

Concentration-Bound Analysis for Probabilistic Programs and Probabilistic Recurrence Relations

JINYI WANG*, Shanghai Jiao Tong University, China

YICAN SUN*, Peking University, China

HONGFEI FU, Shanghai Jiao Tong University, China

MINGZHANG HUANG, IST Austria (Institute of Science and Technology Austria), Austria

AMIR KAFSHDAR GOHARSHADY, IST Austria, Austria

KRISHNENDU CHATTERJEE, IST Austria, Austria

Analyzing probabilistic programs and randomized algorithms are classical problems in computer science. The first basic problem in the analysis of stochastic processes is to consider the expectation or mean, and another basic problem is to consider concentration bounds, i.e. showing that large deviations from the mean have small probability. Similarly, in the context of probabilistic programs and randomized algorithms, the analysis of expected termination time/running time and their concentration bounds are fundamental problems. In this work, we focus on concentration bounds for probabilistic programs and probabilistic recurrences of randomized algorithms. For probabilistic programs, the basic technique to achieve concentration bounds is to consider martingales and apply the classical Azuma's inequality [Azuma 1967]. For probabilistic recurrences of randomized algorithms, Karp's classical "cookbook" method [Karp 1994], which is similar to the master theorem for recurrences, is the standard approach to obtain concentration bounds. In this work, we propose a novel approach for deriving concentration bounds for probabilistic programs and probabilistic recurrence relations through the synthesis of exponential supermartingales. For probabilistic programs, we present algorithms for synthesis of such supermartingales in several cases. We also show that our approach can derive better concentration bounds than simply applying the classical Azuma's inequality over various probabilistic programs considered in the literature. For probabilistic recurrences, our approach can derive tighter bounds than the well-established methods of [Karp 1994] on classical algorithms such as quick sort, quick select, and randomized diameter computation. Moreover, we show that our approach could derive bounds comparable to the optimal bound for quicksort, proposed in [McDiarmid and Hayward 1996]. We also present a prototype implementation that can automatically infer these bounds.

1 INTRODUCTION

In this work, we present new methods to obtain concentration bounds for probabilistic programs and probabilistic recurrences. In this section, we start with a brief description of probabilistic programs and recurrences and provide an overview of the basic analysis problems. Then, we discuss previous results and finally present our contributions.

Probabilistic programs and recurrences. The formal analysis of probabilistic models is a fundamental problem that spans across various disciplines, including probability theory and statistics [Durrett 1996; Howard 1960; Kemeny et al. 1966; Paz 1971; Rabin 1963], randomized algorithms [Motwani and Raghavan 1995], formal methods [Baier and Katoen 2008; Kwiatkowska et al. 2011], artificial intelligence [Kaelbling et al. 1998, 1996], and programming languages [Barthe et al.

*equal contribution

Authors' addresses: Jinyi Wang, Shanghai Jiao Tong University, Shanghai, China, jinyi.wang@sjtu.edu.cn; Yican Sun, Peking University, Beijing, China, sycpku@pku.edu.cn; Hongfei Fu, Shanghai Jiao Tong University, Shanghai, China, fuhf@cs.sjtu.edu.cn; Mingzhang Huang, IST Austria (Institute of Science and Technology Austria), Klosterneuburg, Austria, mingzhang.huang@ist.ac.at; Amir Kafshdar Goharshady, IST Austria, Klosterneuburg, Austria, goharshady@gmail.com; Krishnendu Chatterjee, IST Austria, Klosterneuburg, Austria, krishnendu.chatterjee@ist.ac.at.

2020]. The analysis of probabilistic programs, i.e. imperative programs extended with random value generators, has received significant attention in the programming languages community [Barthe et al. 2018, 2017; Chakarov and Sankaranarayanan 2013; Chatterjee et al. 2016, 2018b; Cusumano-Towner et al. 2018; Dahlqvist and Kozen 2020; Esparza et al. 2012; Fioriti and Hermanns 2015; Foster et al. 2016; Hark et al. 2020; Olmedo et al. 2016; Sankaranarayanan et al. 2013]. Similarly, the analysis of randomized algorithms is a central and classical problem in theoretical computer science [Motwani and Raghavan 1995], and a key problem is to analyze the probabilistic recurrence relations arising from randomized algorithms.

Basic analysis problems. In the analysis of probabilistic programs and probabilistic recurrences, one has to analyze the underlying stochastic processes. The first basic problem in analysis of stochastic processes is to consider the expectation or the mean [Williams 1991a]. However, the expectation (or the first moment) does not provide enough information about the probability distribution associated with the process. Hence higher moments (such as variance) are key parameters of interest. Another fundamental problem is to obtain *concentration bounds* showing that large deviation from the mean has small probability. A key advantage of establishing strong concentration bounds is that it enables us to provide high-probability guarantees (e.g. with high probability the running time does not exceed a desired bound). In the context of probabilistic programs and randomized algorithms, a key quantity of interest is the termination time of programs or the running time associated with probabilistic recurrences. Thus, the analysis of expected termination time/running time and their concentration bounds for probabilistic programs and probabilistic recurrences are fundamental problems in computer science.

Previous results on expectation analysis. The problem of expected termination time analysis has received huge attention. The expected termination time analysis for probabilistic programs has been widely studied with different techniques, e.g. martingale-based techniques [Chakarov and Sankaranarayanan 2013; Fu and Chatterjee 2019] and weakest pre-expectation calculus [Kaminski et al. 2016]. The analysis of expected running time of probabilistic recurrences is a fundamental problem studied in randomized algorithms [Motwani and Raghavan 1995], and automated methods for them have also been considered [Chatterjee et al. 2017].

Previous results on concentration bounds. The analysis of concentration bounds is more involved than expectation analysis, both for the general case of stochastic processes, as well as in the context of probabilistic programs and probabilistic recurrences. For probabilistic programs, the only technique in the literature to obtain concentration bounds is to consider either linear or polynomial supermartingales [Chakarov and Sankaranarayanan 2013; Chatterjee et al. 2016], and then apply the classical Azuma's inequality [Azuma 1967] on the supermartingales to obtain concentration bounds. For probabilistic recurrences of randomized algorithms, the standard approach for concentration bounds is Karp's classical "cookbook" method [Karp 1994], which is similar to the master theorem for non-probabilistic recurrences. More advanced methods have also been developed for specific algorithms such as QUICKSORT [Dubhashi and Panconesi 2009; McDiarmid and Hayward 1996].

Our contributions. In this work, we consider the problem of concentration bounds for probabilistic programs and probabilistic recurrences and our main contributions are as follows:

- (1) First, we propose a novel approach for deriving concentration bounds for probabilistic programs and probabilistic recurrence relations through the synthesis of exponential supermartingales.
- (2) For probabilistic programs, we present algorithms for the synthesis of such supermartingales for several cases. We show that our approach can derive better concentration bounds than simply applying the classical Azuma's inequality, over various probabilistic programs considered in the literature.

- (3) For probabilistic recurrences, we show that our approach can derive tighter bounds than the classical methods of the literature on several basic problems. We show that our concentration bounds for probabilistic recurrences associated with classical randomized algorithms such as QUICKSELECT (which generalizes median selection), QUICKSORT and randomized diameter computation beat the bounds obtained using methods of [Karp 1994]. Moreover, we show that our approach could derive bounds comparable to the optimal bound for quicksort, proposed in [McDiarmid and Hayward 1996]. We also present a prototype implementation that can automatically infer these bounds.

Novelty. The key novelty of our approach is as follows: Instead of linear or polynomial martingales, we consider exponential martingales, and our concentration bounds are obtained using the standard Markov's inequality. This is quite a simple technique. The surprising and novel aspect is that with such a simple technique, we can (a) obtain better concentration bounds for probabilistic programs in comparison with the only existing method of applying Azuma's inequality; and (b) improve the more than two-decades old classical bounds for basic and well-studied randomized algorithms.

2 PRELIMINARIES

Throughout the paper, we denote by \mathbb{N} , \mathbb{N}_0 , \mathbb{Z} , and \mathbb{R} the sets of positive integers, non-negative integers, integers, and real numbers, respectively. We first review several fundamental concepts in probability theory and then illustrate the problems to study.

2.1 Basics of Probability Theory

We provide a short review of some necessary concepts in probability theory. For a more detailed treatment, see [Williams 1991a].

Probability Distributions. A discrete probability distribution over a countable set U is a function $p : U \rightarrow [0, 1]$ such that $\sum_{u \in U} p(u) = 1$. The support of p is defined as $\text{supp}(p) := \{u \in U \mid p(u) > 0\}$.

Probability Spaces. A probability space is a triple $(\Omega, \mathcal{F}, \text{Pr})$, where Ω is a non-empty set (called the sample space), \mathcal{F} is a σ -algebra over Ω (i.e. a collection of subsets of Ω that contains the empty set \emptyset and is closed under complementation and countable union) and Pr is a probability measure on \mathcal{F} , i.e. a function $\text{Pr} : \mathcal{F} \rightarrow [0, 1]$ such that (i) $\text{Pr}(\Omega) = 1$ and (ii) for all set-sequences $A_1, A_2, \dots \in \mathcal{F}$ that are pairwise-disjoint (i.e. $A_i \cap A_j = \emptyset$ whenever $i \neq j$) it holds that $\sum_{i=1}^{\infty} \text{Pr}(A_i) = \text{Pr}(\bigcup_{i=1}^{\infty} A_i)$. Elements of \mathcal{F} are called events. An event $A \in \mathcal{F}$ holds almost-surely (a.s.) if $\text{Pr}(A) = 1$.

Random Variables. A random variable X from a probability space $(\Omega, \mathcal{F}, \text{Pr})$ is an \mathcal{F} -measurable function $X : \Omega \rightarrow \mathbb{R} \cup \{-\infty, +\infty\}$, i.e. a function such that for all $d \in \mathbb{R} \cup \{-\infty, +\infty\}$, the set $\{\omega \in \Omega \mid X(\omega) < d\}$ belongs to \mathcal{F} .

Expectation. The expected value of a random variable X from a probability space $(\Omega, \mathcal{F}, \text{Pr})$, denoted by $\mathbb{E}(X)$, is defined as the Lebesgue integral of X w.r.t. Pr , i.e. $\mathbb{E}(X) := \int X \, d\text{Pr}$. The precise definition of Lebesgue integral is somewhat technical and is omitted here (cf. [Williams 1991a, Chapter 5] for a formal definition). If range $X = \{d_0, d_1, \dots\}$ is countable, then we have $\mathbb{E}(X) = \sum_{k=0}^{\infty} d_k \cdot \text{Pr}(X = d_k)$.

Filtrations. A filtration of a probability space $(\Omega, \mathcal{F}, \text{Pr})$ is an infinite sequence $\{\mathcal{F}_n\}_{n \in \mathbb{N}_0}$ of σ -algebras over Ω such that $\mathcal{F}_n \subseteq \mathcal{F}_{n+1} \subseteq \mathcal{F}$ for all $n \in \mathbb{N}_0$. Intuitively, a filtration models the information available at any given point of time.

Conditional Expectation. Let X be any random variable from a probability space $(\Omega, \mathcal{F}, \text{Pr})$ such that $\mathbb{E}(|X|) < \infty$. Then, given any σ -algebra $\mathcal{G} \subseteq \mathcal{F}$, there exists a random variable (from $(\Omega, \mathcal{F}, \text{Pr})$), denoted by $\mathbb{E}(X|\mathcal{G})$, such that:

- (E1) $\mathbb{E}(X|\mathcal{G})$ is \mathcal{G} -measurable, and

(E2) $\mathbb{E}(|\mathbb{E}(X|\mathcal{G})|) < \infty$, and

(E3) for all $A \in \mathcal{G}$, we have $\int_A \mathbb{E}(X|\mathcal{G}) d\Pr = \int_A X d\Pr$.

The random variable $\mathbb{E}(X|\mathcal{G})$ is called the *conditional expectation* of X given \mathcal{G} . The random variable $\mathbb{E}(X|\mathcal{G})$ is a.s. unique in the sense that if Y is another random variable satisfying (E1)–(E3), then $\Pr(Y = \mathbb{E}(X|\mathcal{G})) = 1$. We refer to [Williams 1991a, Chapter 9] for details. Intuitively, $\mathbb{E}(X|\mathcal{G})$ is the expectation of X , when assuming the information in \mathcal{G} .

Stochastic Processes. A (discrete-time) stochastic process is a sequence $\Gamma = \{X_n\}_{n \in \mathbb{N}_0}$ of random variables where X_n 's are all from some probability space $(\Omega, \mathcal{F}, \Pr)$. The process Γ is *adapted* to a filtration $\{\mathcal{F}_n\}_{n \in \mathbb{N}_0}$ if for all $n \in \mathbb{N}_0$, X_n is \mathcal{F}_n -measurable. Intuitively, the random variable X_i models some value at the i -th step of the process.

Martingales and Supermartingales. A stochastic process $\Gamma = \{X_n\}_{n \in \mathbb{N}_0}$ adapted to a filtration $\{\mathcal{F}_n\}_{n \in \mathbb{N}_0}$ is a *martingale* (resp. *supermartingale*) if for every $n \in \mathbb{N}_0$, $\mathbb{E}(|X_n|) < \infty$ and it holds a.s. that $\mathbb{E}(X_{n+1}|\mathcal{F}_n) = X_n$ (resp. $\mathbb{E}(X_{n+1}|\mathcal{F}_n) \leq X_n$). We refer to [Williams 1991a, Chapter 10] for a deeper treatment.

Intuitively, a martingale (resp. supermartingale) is a discrete-time stochastic process in which for an observer who has seen the values of X_0, \dots, X_n , the expected value at the next step, i.e. $\mathbb{E}(X_{n+1}|\mathcal{F}_n)$, is equal to (resp. no more than) the last observed value X_n . Also, note that in a martingale, the observed values for X_0, \dots, X_{n-1} do not matter given that $\mathbb{E}(X_{n+1}|\mathcal{F}_n) = X_n$. In contrast, in a supermartingale, the only requirement is that $\mathbb{E}(X_{n+1}|\mathcal{F}_n) \leq X_n$ and hence $\mathbb{E}(X_{n+1}|\mathcal{F}_n)$ may depend on X_0, \dots, X_{n-1} . Also, note that \mathcal{F}_n might contain more information than just the observations of X_i 's.

Stopping Times. Given a probability space $(\Omega, \mathcal{F}, \Pr)$, a *stopping time* w.r.t a filtration $\{\mathcal{F}_n\}_{n \in \mathbb{N}_0}$ is a random variable $T : \Omega \rightarrow \mathbb{N}_0 \cup \{\infty\}$ such that $\{\omega \mid T(\omega) = n\} \in \mathcal{F}_n$.

Example 2.1. Consider an unbiased and discrete random walk, in which we start at a position X_0 , and at each second walk one step to either left or right with equal probability. Let X_n denote our position after n seconds. It is easy to verify that $\mathbb{E}(X_{n+1}|X_0, \dots, X_n) = \frac{1}{2}(X_n - 1) + \frac{1}{2}(X_n + 1) = X_n$. Hence, this random walk is a martingale. Note that by definition, every martingale is also a supermartingale. As another example, consider the classical gambler's ruin: a gambler starts with Y_0 dollars of money and bets continuously until he loses all of his money. If the bets are unfair, i.e. the expected value of his money after a bet is less than its expected value before the bet, then the sequence $\{Y_n\}_{n \in \mathbb{N}_0}$ is a supermartingale. In this case, Y_n is the gambler's total money after n bets. On the other hand, if the bets are fair, then $\{Y_n\}_{n \in \mathbb{N}_0}$ is a martingale.

2.2 Termination and Concentration Bounds

In this work, we consider concentration bounds of the termination time of probabilistic programs and recurrence relations. Below, we illustrate the notions of termination time and concentration bound.

Termination Time. The termination-time random variable counts the number of steps a program takes to termination. In the setting of probabilistic programs, the termination-time random variable is defined as the first time the program hits its termination program counter, which is a stopping time. For convenience, we treat each termination-time random variable as the total accumulated cost until the program terminates, for which each execution step of the program causes a single unit of cost. In the setting of probabilistic recurrence relations, since the amount of steps in a preprocessing stage is directly added to the termination time, we treat the number of steps in a preprocessing stage as the cost at current execution step, and define the termination-time variable as the total accumulated cost until the recurrence relation runs into the empty stack (i.e., terminates).

In both cases, we consider the termination time random variable as the total accumulated cost until the stopping time of termination.

Concentration Bounds. A *concentration bound* of a random variable C is an inequality of the form $\Pr(C \geq d) \leq f(d)$ or $\Pr(C \leq d) \leq f(d)$ which specifies whether the probability that the value of C grows too large or too small is bounded by a function f , which tends to zero when $d \rightarrow \infty$ (for $\Pr(C \geq d)$) or $d \rightarrow -\infty$ (for $\Pr(C \leq d)$). Concentration bounds are an important probabilistic property of random variables as they witness that a random variable will have a substantially small probability of large deviation. When applied to termination time, the concentration bound specifies the following property: the probability that a randomized algorithm runs inefficiently, i.e. takes much longer than its expected termination time, is very small. Thus compared with expected termination time, concentration bounds provide a much finer characterization of the running time of randomized algorithms.

Problem Statement. In this work, we consider the following problems: For probabilistic programs, we consider unnested probabilistic while loops. Our aim is to have automated approaches for deriving much tighter exponentially-decreasing concentration bounds in comparison with the existing approach through Azuma or Hoeffding inequalities [Chatterjee et al. 2018b]. For probabilistic recurrence relations, our goal is to derive substantially finer bounds compared with the classical results of [Karp 1994], and for quicksort, we try to derive bounds comparable to the optimal result proposed in [McDiarmid and Hayward 1996]. Obtaining more precise (tighter) concentration bounds is important in many applications, e.g. resource-contention resolution [Kleinberg and Tardos 2006, Chapter 13], or in resource-constrained environments such as embedded systems.

3 THE GENERAL METHOD

In this section, we establish the mathematical foundation for the analysis of concentration bounds, based on which we will later develop novel sound approaches to derive concentration bounds for the termination time arising from probabilistic programs and recurrence relations. As in certain situations, e.g. probabilistic recurrence relations, the termination time is defined as the total cost accumulated at every execution step, we consider concentration bounds for a general non-negative random variable. First, we introduce the basic result through which we derive our concentration bounds. This lemma is similar to the basis for Chernoff bound.

LEMMA 3.1. *For any real number d , random variable C and $\beta > 1$, we have $\Pr(C \geq d) \leq \mathbb{E}(\beta^C)/\beta^d$.*

PROOF. Since $\beta > 1$, we have $\Pr(C \geq d) = \Pr(\beta^C \geq \beta^d)$. Then by applying Markov's inequality, we obtain the desired result. \square

Lemma 3.1 provides a mathematical way to derive a concentration bound for any random variable C , but yet we do not know an upper bound for $\mathbb{E}(\beta^C)$. In the setting of Chernoff bounds, one often chooses to evaluate $\mathbb{E}(\beta^C)$, e.g. when the random variable C is a finite sum of independent random variables. In our setting, the situation is more complicated, because we consider C to be the total accumulated cost until a stopping time*. We use supermartingales and Optional Stopping Theorem [Williams 1991a, Chapter 10.10] to derive an upper bound for $\mathbb{E}(\beta^C)$, as is shown by the following proposition. In the proposition below, $\{C_n\}_{n \in \mathbb{N}_0}$ is a stochastic process where each C_n represents the amount of cost accumulated at the n -th step so that $C = \sum_{n=0}^{T-1} C_n$, and $\{X_n\}_{n \in \mathbb{N}_0}$ is a key stochastic process for deriving an upper bound for $\mathbb{E}(\beta^C)$.

PROPOSITION 3.2. *Consider two stochastic processes $\{X_n\}_{n \in \mathbb{N}_0}$ and $\{C_n\}_{n \in \mathbb{N}_0}$ adapted to a filtration $\{\mathcal{F}_n\}$ for which X_0 is a constant random variable, and a stopping time T w.r.t the filtration $\{\mathcal{F}_n\}$. Let $\alpha, \beta > 1$ be real numbers. If it holds that*

*If the cost accumulated at each step is equal to one, then the total accumulated cost is equal to the running time.

- (C1) : T is a.s. finite, i.e., $\Pr(T < \infty) = 1$, and
- (C2) : $X_T \geq K$ a.s. for some constant $K \leq 0$, and
- (C3) : $\beta^{C_n} \cdot \mathbb{E}(\alpha^{X_{n+1}} | \mathcal{F}_n) \leq \alpha^{X_n}$ a.s. for all $n \in \mathbb{N}_0$,

then we have $\mathbb{E}(\beta^C) \leq \alpha^{X_0-K}$ where $C := \sum_{n=0}^{T-1} C_n$. Here, the existence and (probabilistic) uniqueness of the conditional expectation $\mathbb{E}(\alpha^{X_{n+1}} | \mathcal{F}_n)$ is due to its non-negativity (see [Agrawal et al. 2018, Proposition 3.1]).

PROOF. Define the stochastic process $\{Y_n\}_{n \geq 0}$ as $Y_n := \alpha^{X_n} \cdot \beta^{\sum_{j=0}^{n-1} C_j}$. By definition, we have that $Y_n > 0$ for all n . Note that although Y_n may be non-integrable (due to its exponential construction), its conditional expectation $\mathbb{E}(Y_n | \mathcal{F}_n)$ still exists, following from $Y_n > 0$ and [Agrawal et al. 2018, Proposition 3.1]. By the condition (C3) and the “take out what is known” property of conditional expectation (see [Williams 1991a, Chapter 9.7]), we have that $\mathbb{E}(\alpha^{X_{n+1}} \cdot \beta^{\sum_{j=0}^n C_j} | \mathcal{F}_n) \leq \alpha^{X_n} \cdot \beta^{\sum_{j=0}^{n-1} C_j}$. (Note that although the integrability may not be ensured, but the proof on [Williams 1991a, Page 90] guarantees the case for non-negative random variables.) Then we have that $\mathbb{E}(Y_{n+1} | \mathcal{F}_n) \leq Y_n$ a.s., which means $\mathbb{E}(Y_{n+1}) \leq \mathbb{E}(Y_n)$ from the basic property of conditional expectation. It follows from an easy induction on n that $\mathbb{E}(Y_n) \leq \mathbb{E}(Y_0) < \infty$ for all $n \geq 0$, thus the conditional expectation is also taken in the normal sense as each Y_n is indeed integrable, and $\{Y_n\}_{n \in \mathbb{N}_0}$ is then a supermartingale. Moreover, the process $\{Y_n\}_{n \geq 0}$ is a non-negative supermartingale by definition. Then by applying Optional Stopping Theorem [Williams 1991a, Chapter 10.10] and using (C1), we obtain that $\mathbb{E}(Y_T) \leq \mathbb{E}(Y_0) = \alpha^{X_0}$. Now, from the condition (C2), we have $Y_T \geq \alpha^K \cdot \beta^C$ a.s. It follows that $\mathbb{E}(\beta^C) \leq \alpha^{X_0-K}$. \square

Proposition 3.2 provides a way to bound $\mathbb{E}(\beta^C)$ by a stochastic process $\{X_n\}_{n \geq 0}$ and an auxiliary $\alpha > 1$ if the conditions (C1)–(C3) are satisfied. By combining Lemma 3.1 and Proposition 3.2, we obtain the following main theorem of this section for concentration bounds of total accumulated cost until a stopping time.

THEOREM 3.3. *Let $\{X_n\}_{n \in \mathbb{N}_0}$ and $\{C_n\}_{n \in \mathbb{N}_0}$ be stochastic processes adapted to a filtration $\{\mathcal{F}_n\}$ for which X_0 is a constant random variable, and T be a stopping time with respect to the filtration $\{\mathcal{F}_n\}$. Let $\alpha, \beta > 1$ be real numbers. If the conditions (C1)–(C3) (cf. Proposition 3.2) are fulfilled (by X_n, C_n ’s, α, β and T), then we have that $\Pr(C \geq d) \leq \alpha^{X_0-K} \cdot \beta^{-d}$ for every $d \in \mathbb{R}$, where $C := \sum_{n=0}^{T-1} C_n$.*

PROOF. By Lemma 3.1, we have $\Pr(C \geq d) \leq \mathbb{E}(\beta^C) / \beta^d$. By Proposition 3.2, we have $\mathbb{E}(\beta^C) \leq \alpha^{X_0-K}$. Thus, we have $\Pr(C \geq d) \leq \alpha^{X_0-K} \cdot \beta^{-d}$. \square

REMARK 1 (COMPARISON WITH CLASSICAL METHODS). *Compared with classical mathematical methods such as Chernoff bounds and Hoeffding’s inequality that only examine the expected value, variance or range of related random variables, our method (Theorem 3.3) examines the detailed probability distribution by deriving a better bound for the moment generation function $\mathbb{E}(\beta^C)$. The derivation is through the construction of an exponential supermartingale (see Proposition 3.2), and depends on the detailed probability distributions in the considered problem. Since we examine the probability distributions in detail, our method can obtain much tighter concentration bound than the inequalities of Azuma and Hoeffding [Chatterjee et al. 2018b] and Karp’s classical method [Karp 1994]. We do not consider Chernoff bounds as they are typically used for a finite sum of independent random variables, while our method focuses on concentration bounds for total accumulated costs until a stopping time. We show how the method can be automated for several classical and widely-used probability distributions, such as uniform and Bernoulli.*

4 CONCENTRATION BOUNDS FOR PROBABILISTIC PROGRAMS

In this section, we apply our method (Theorem 3.3) to probabilistic programs. Probabilistic programs are imperative programs extended with random number generators. Here we consider the concentration bound of a single probabilistic while loop in which there is no nested loops. We first illustrate how we can apply our method to a single probabilistic while loop. Then we develop automated algorithms for deriving concentration bound. Finally, we present experimental results and compare our results with previous methods.

4.1 The General Method Applied to Probabilistic While Loops

We consider a probabilistic while loop of the form

$$\text{while } G \text{ do } Q \text{ od} \quad (1)$$

where G is the loop guard and Q an imperative probabilistic program with possibly assignment statements with samplings, conditional branches, probabilistic branches, but without (nested) while loops. For a detailed treatment of the syntax, we refer to [Chatterjee et al. 2018b]. Since we only consider single probabilistic while loops here, we choose to have a light-weight description of probabilistic programs.

Before we describe probabilistic programs, we first illustrate a simple example.

Example 4.1. Consider the probabilistic while loop:

while $(x \geq 0)$ **do** $x := x + r$ **od**

In each loop iteration, the value of the variable x is added a random sampling r whose distribution is given by $\Pr(r = -1) = 0.75$, $\Pr(r = 1) = 0.25$. The random value of r is sampled independently in every iteration. The loop ends when the value of x falls below zero.

Below we describe the behaviours of a probabilistic while loop. In the sequel, we fix two disjoint sets of variables: the set V_p of *program variables* and the set V_r of *sampling variables*. Informally, program variables are real-valued and are directly related to the control flow of a program, while sampling variables represent random inputs sampled from their predefined probability distributions.

Valuations. A *valuation* over a finite set V of variables is a function $v : V \rightarrow \mathbb{R}$ that assigns a real value to each variable. We denote by Val_V the set of all valuations over V . For simplicity, we treat each loop guard G of a probabilistic while loop as a subset of Val_{V_p} so that a valuation satisfies the loop guard iff the valuation falls in the designated subset G . Moreover, we may also regard each element $v \in Val_{V_p}$ (resp. $\mu \in Val_{V_r}$) as a vector v (resp. μ) with an implicit ordering between program variables (resp. sampling variables), such that $v[i]$ (resp. $\mu[i]$) represents the value of the i th program variable (resp. sampling variable), respectively.

The Semantics. We present a lightweight semantics for probabilistic while loops. We describe the execution of a probabilistic while loop by an infinite sequence of vectors of random variables $\{v_n\}_{n \geq 0}$ inductively defined as follows. Below we fix a probabilistic while loop in the form (1).

- v_0 is a vector of constant random variables representing the initial input;
- if $v \in G$ (i.e., v satisfies the loop guard), then $v_{n+1} = F(v_n, \mu_n)$, where (i) the vector μ_n represents the sampled values for sampling variables at the $(n + 1)$ th loop iteration, and (ii) F is the *update function* for the loop body such that given the current valuation $v \in Val_{V_p}$ for the program variables before the current loop iteration and the values $\mu \in Val_{V_r}$ sampled in the current loop iteration, $F(v, \mu)$ is the updated vector for program variables after the execution of the current loop iteration;
- if $v \notin G$ (i.e., v does not satisfy the loop guard), then $v_{n+1} = v_n$.

The inductive definition can be turned into a general state-space Markov chain by defining suitable kernel functions for the update function F , see [Meyn and Tweedie 1993, Chapter 3].

In the paper, we consider the simplified setting that the sampled values are discrete with bounded range. For this simplified setting, we use a *sampling function* Υ to describe the sampling, which assigns to every sampling variable $r \in V_r$ the predefined discrete probability distribution over \mathbb{R} . Then, the *joint* discrete probability distribution $\bar{\Upsilon}$ over Val_{V_r} is defined as $\bar{\Upsilon}(\mu) := \prod_{r \in V_r} \Upsilon(r)(\mu(r))$ for all valuations μ over sampling variables.

Below we apply our method (Theorem 3.3) to probabilistic while loops, where we consider that the cost C_n to execute the $(n + 1)$ th loop iteration is equal to 1, so that the total accumulated cost is equal to the number of loop iterations of the loop. Recall that F is the update function for the loop body.

THEOREM 4.2 (CONCENTRATION FOR PROBABILISTIC PROGRAMS). *Let P be a probabilistic while loop in the form (1) and T be the random variable representing the number of loop iterations of P until termination. Suppose that (i) T is a.s. finite (i.e., P terminates with probability one) and (ii) there exist real numbers $\alpha, \beta > 1, K \leq 0$ and a Borel measurable function $\eta : \text{Val}_{V_p} \rightarrow \mathbb{R}$ such that*

- (A1) : $\eta(v) \geq K$ for all $v \in G$;
- (A2) : $\eta(v, \mu) \geq K$ for all $v \in G$ and $\mu \in V_r$;
- (A3) : $\beta \cdot \sum_{\mu \in \text{Val}_{V_r}} \bar{\Upsilon}(\mu) \alpha^{\eta(F(v, \mu))} \leq \alpha^{\eta(v)}$ for every $v \in G$.

Then for any initial valuation $v_0 \in G$, we have $\Pr(T \geq n) \leq \alpha^{\eta(v_0) - K} \cdot \beta^{-n}$ for all $n \geq 0$.

PROOF. Choose $C_n = 1$ for every $n \geq 0$. Then $T = C = \sum_{n=0}^{T-1} C_n$. Let v_n be the vector of random variables representing the valuation for program variables at the n th step (where v_0 is a vector of constant random variables that represents the initial input). Define the filtration $\{\mathcal{F}_n\}_{n \geq 0}$ for which \mathcal{F}_n is the smallest sigma-algebra that makes all random variables in v_0, \dots, v_n measurable. Define the stochastic process $X_n := \eta(v_n)$ for $n \geq 0$. From the first two conditions, we have that $X_n \geq K$ for all n . Note that the second condition specifies that the value of the process is no less than K at the point the loop terminates. Then from the third condition and the “role of independence” property on [Williams 1991b, Page 90], we have that the condition (C3) holds. Thus, by Theorem 3.3, we have $\Pr(T \geq n) \leq \alpha^{X_0 - K} / \beta^n$. \square

4.2 A Synthesis Algorithm for Exponential Supermartingales

Reasoning about exponential terms is an inherently challenge for algorithmic analysis. Here we provide a practical synthesis algorithm for exponential supermartingales in Theorem 4.2 through ranking-supermartingale (RSM) synthesis. Our heuristics is that we treat the function η as an RSM-map, which is the core in existing RSM-synthesis algorithms. Then based on the synthesized RSM-map η , we resolve β, α through polynomial solvers such as MATLAB.

The Loops We Consider. We consider that every assignment in the loop is *incremental*, i.e., (i) every assignment has the form $x := x + e$ where x is a program variable and e is a linear expression consisting of constants and sampling variables. Although incremental assignments are in limited form, they are expressive enough to implement random walks, gambler’s ruin and many other examples in the literature. Since we utilize the synthesized linear RSMs, we also assume that the loop guard G can be expressed as a polyhedron (i.e., a conjunction of linear inequalities). Also recall that we only handle samplings that observes discrete probability distributions with bounded range.

Before we illustrate the synthesis algorithm, we first recall the notion of RSM-maps, where we adopt the simplified version for a single probabilistic loop. Below, we fix an input probabilistic while loop P in the form (1). Recall that F is the update function.

Definition 4.3 (RSM-maps). An RSM-map is a Borel measurable function $\eta : \text{Val}_{V_p} \rightarrow \mathbb{R}$ such that there exist constants $\epsilon > 0, K, K' \leq 0$ such that

- (B1) : $\eta(v) \geq 0$ for all $v \in G$;
- (B2) : $K \leq \eta(v, \mu) \leq K'$ for all $v \in G$ and $\mu \in V_r$;
- (B3) : $\sum_{\mu \in Val_{V_r}} \bar{Y}(\mu) \eta(F(v, \mu)) \leq \eta(v) - \epsilon$ for every $v \in G$.

Our algorithm fixes the ϵ to be 1 in the definition as the function η can be scaled by any factor. Moreover, in our algorithm, the constant K will correspond to the same K in (A1) and (A2). We do not use K' in our algorithm.

The Synthesis Algorithm. The algorithm first synthesizes a linear RSM-map η through existing algorithms (which is not the focus of this paper). Then the algorithm searches the value for β through a binary search. Once a value for β is chosen, we solve the minimum value for α (so as to obtain the best bound from Theorem 3.3) through the MATLAB polynomial solver. A key point here is that since we only consider incremental assignments and the RSM-map η is linear, the condition (A3) reduces to a polynomial inequality that only involves α when the value for β is chosen. The algorithm thus proceeds by searching larger and larger β through binary search until a given precision is reached, and then finds the corresponding minimal α . Then we apply Theorem 4.2 to obtain the concentration bound.

4.3 Prototype Implementation and Experimental Results

Implementation. We implemented the algorithm of Section 4.2 using Matlab. All results were obtained on a machine with an Intel Core i7-8700 processor (3.2 GHz) and 16 GB of memory, running Microsoft Windows 10.

Benchmarks. We use benchmarks from [Chatterjee et al. 2018a,b; Ngo et al. 2018]. Our benchmarks are as follows:

- **AdvRW1D:** This program models a 1-dimensional adversarial random walk and is taken from [Chatterjee et al. 2018b].
- **RDWALK1, RDWALK2, RDWALK3:** These programs model skewed 1-dimensional random walks, in which the probabilities of going to the left and to the right at each step are not equal. They are taken from [Ngo et al. 2018]. In RDWALK1 there is a 0.75 probability of moving left and a 0.25 probability of moving right. In RDWALK2 the probabilities are 0.875, 0.125, and in RDWALK3, they are 0.9375, 0.0625.
- **PRSPED:** This example is also a random walk taken from [Ngo et al. 2018]. The main difference is that the step length is not necessarily 1.
- **MINI-ROU, MINI-ROU2:** These programs are taken from [Chatterjee et al. 2018a]. They are implementations of the mini-roulette casino game as probabilistic programs. MINI-ROU is the benchmark in [Chatterjee et al. 2018a], while MINI-ROU2 is a variant where the probabilities of losing gambles are increased in order to reduce the runtime.

Results and Discussion. Our experimental results are presented in Table 1. We compare our concentration bounds with those obtained using Azuma's inequality and Hoeffding's inequality in [Chatterjee et al. 2018b]. As shown in Table 1 our approach successfully obtains tighter bounds in every case. Moreover, note that our approach is very efficient and can obtain these bounds in a few seconds.

Table 1. Our Experimental Results over Probabilistic Programs.

Program	α	β	κ	Our bound for $P(T > \kappa)$	[Chatterjee et al. 2018b]'s bound for $P(T > \kappa)$	RSM (η)	Our runtime (s)
vad1D	1.1915	1.1003	$6X_0$	2.25×10^{-1}	8.16×10^{-1}	2.8571x	4.68
			$10X_0$	3.33×10^{-2}	5.12×10^{-1}		
			$14X_0$	4.90×10^{-3}	3.07×10^{-1}		
			$18X_0$	7.28×10^{-4}	1.81×10^{-1}		
			$22X_0$	1.08×10^{-4}	1.06×10^{-1}		
rdwalk1	1.314	1.1547	$6X_0$	9.07×10^{-2}	2.10×10^{-1}	2x	3.82
			$10X_0$	5.11×10^{-3}	2.06×10^{-2}		
			$14X_0$	2.88×10^{-4}	1.82×10^{-3}		
			$18X_0$	1.62×10^{-6}	1.56×10^{-4}		
			$22X_0$	9.13×10^{-7}	1.31×10^{-5}		
rdwalk2	2.072	1.511	$6X_0$	4.16×10^{-4}	7.92×10^{-3}	$\frac{3}{4}x$	4.34
			$10X_0$	1.07×10^{-7}	3.41×10^{-5}		
			$14X_0$	2.75×10^{-11}	1.32×10^{-7}		
			$18X_0$	7.05×10^{-15}	4.97×10^{-10}		
			$22X_0$	1.81×10^{-18}	1.84×10^{-12}		
rdwalk3	3.265	2.065	$6X_0$	5.07×10^{-7}	7.79×10^{-4}	1.142x	5.74
			$10X_0$	2.54×10^{-13}	4.39×10^{-7}		
			$14X_0$	1.27×10^{-19}	2.24×10^{-10}		
			$18X_0$	6.35×10^{-26}	1.10×10^{-13}		
			$22X_0$	3.18×10^{-32}	5.35×10^{-17}		
mini-Rou	1.00124	1.00068	$6X_0$	1.00×10^{-1}	9.96×10^{-1}	13x – 13	7.02
			$10X_0$	9.88×10^{-1}	9.99×10^{-1}		
			$14X_0$	9.74×10^{-1}	1.00×10^{-1}		
			$18X_0$	9.96×10^{-1}	9.99×10^{-1}		
			$22X_0$	9.48×10^{-1}	9.98×10^{-1}		
mini-Rou2	1.607	1.309	$6X_0$	3.90×10^{-3}	1.44×10^{-1}	0.323x – 0.323	6.68
			$10X_0$	1.77×10^{-5}	3.26×10^{-2}		
			$14X_0$	8.11×10^{-8}	7.30×10^{-3}		
			$18X_0$	3.71×10^{-10}	1.60×10^{-3}		
			$22X_0$	1.69×10^{-12}	3.69×10^{-4}		
prspeed	2.0479	1.3048	$6X_0$	4.90×10^{-1}	2.52×10^{-2}	1714 – 1.714x	4.58
			$10X_0$	2.40×10^{-4}	6.00×10^{-3}		
			$14X_0$	1.17×10^{-5}	1.37×10^{-3}		
			$18X_0$	5.71×10^{-8}	3.07×10^{-4}		
			$22X_0$	2.79×10^{-10}	6.84×10^{-5}		

5 CONCENTRATION BOUNDS FOR PROBABILISTIC RECURRENCES

5.1 Problem Setting and Examples

PRRs. In this section, we consider general Probabilistic Recurrence Relations (PRRs) as defined in [Karp 1994]. A PRR is a relation of the following form:

$$\forall n > 1, \mathcal{T}(n) = a(n) + \sum_{i=1}^{\ell} \mathcal{T}(h_i(n)) \quad \mathcal{T}(1) = 0,$$

in which $a(n)$ is a non-negative value and each $h_i(n)$ is a non-negative random variable such that we always have $\sum_{i=1}^{\ell} h_i(n) \leq n - 1$. Intuitively, we think of $\mathcal{T}(n)$ as the time it takes for a randomized divide-and-conquer algorithm to solve an instance of size n . The algorithm first performs a preprocessing procedure, which leads to ℓ smaller instances of random sizes $h_1(n), \dots, h_{\ell}(n)$. It then solves the smaller instances recursively and merges the solutions in a postprocessing phase. We assume that the preprocessing and postprocessing on an instance of size n take $a(n)$ units of time overall. Our goal is to obtain upper bounds on the tail probability $\Pr[\mathcal{T}(n^*) \geq \kappa]$, where n^* is the size of our initial input.

Example 5.1 (QUICKSORT [Hoare 1961a]). One of the most well-known sorting algorithms is QUICKSORT. Given an array with n distinct elements, QUICKSORT first chooses an element p of the array uniformly at random. It then compares all the other $n - 1$ elements of the array with p and divides them in two parts: (i) elements that are smaller than p , and (ii) elements that are larger than p . Finally, it recursively sorts each part. Let $\mathcal{T}(n)$ be the total number of comparisons made by QUICKSORT in handling an array of size n . Assuming that there are $h_1(n)$ elements in part (i) and $h_2(n)$ elements in part (ii) above, we have:

$$\mathcal{T}(n) = n - 1 + \mathcal{T}(h_1(n)) + \mathcal{T}(h_2(n)).$$

Moreover, it is easy to see that $h_1(n) + h_2(n) = n - 1$.

Example 5.2 (QUICKSELECT [Hoare 1961b]). Consider the problem of finding the d -th smallest element in an unordered array of n distinct items. A classical algorithm for solving this problem is QUICKSELECT. Much like QUICKSORT, QUICKSELECT begins by choosing an element p of the array uniformly at random. It then compares all the other $n - 1$ elements of the array with p and divides them into two parts: (i) those that are smaller than p , and (ii) those that are larger. Suppose that there are d' elements in part (i). If $d' < d - 1$, then the algorithm recursively searches for the $(d - d' - 1)$ -th smallest element of part (ii). If $d' = d - 1$, the algorithm terminates by returning p as the desired answer. Finally, if $d' > d - 1$, the algorithm recursively finds the d -th smallest element in part (i). Note that the classical median selection algorithm is a special case of QUICKSELECT. While more involved linear-time non-randomized algorithms exist for the same problem, QUICKSELECT provides a simple randomized variant. In this case, we have the following PRR:

$$\mathcal{T}(n) = n - 1 + \mathcal{T}(h(n)).$$

Here, $\mathcal{T}(n)$ is the number of comparisons performed by QUICKSELECT over an input of size n , and $h(n)$ is a random variable that captures the size of the remaining array that has to be searched recursively.

5.2 Modeling and Theoretical Results

The Markov Chain of a PRR. In order to apply our general approach to PRRs, we first need to embed them in stochastic processes. Suppose we are given a PRR of the following form:

$$\forall n > 0, \mathcal{T}(n) = a(n) + \sum_{i=1}^{\ell} \mathcal{T}(h_i(n)) \quad \mathcal{T}(1) = 0. \quad (2)$$

Moreover, assume that we are interested in $\mathcal{T}(n^*)$ for a specific initial value n^* . We model this PRR as a Markov chain $(X_m)_{m \geq 0}$ in which each state X_i consists of a non-negative integer k_i , and a stack of k_i additional non-negative integers. Formally, for every i , we have $X_i = (k_i, \langle n_1^{(i)}, n_2^{(i)}, \dots, n_{k_i}^{(i)} \rangle)$. Intuitively, each X_i models a state of our probabilistic divide-and-conquer algorithm in which there are k_i more recursive calls waiting to be executed, and the j -th call needs to process an instance of size $n_j^{(i)}$. Following this intuition, the transitions of $(X_m)_{m \geq 0}$ are defined as follows:

$$X_m = \begin{cases} (1, \langle n^* \rangle) & m = 0 \\ (k_{m-1} - 1, \langle n_2^{(m-1)}, \dots, n_{k_{m-1}}^{(m-1)} \rangle) & m > 0 \wedge n_1^{(m-1)} = 1 \\ (k_{m-1} + \ell - 1, \langle h_1(n_1^{(m-1)}), \dots, h_{\ell}(n_1^{(m-1)}), n_2^{(m-1)}, \dots, n_{k_{m-1}}^{(m-1)} \rangle) & m > 0 \wedge n_1^{(m-1)} > 1 \end{cases}$$

Basically, we start with the state $(1, \langle n^* \rangle)$. In other words, in the beginning, we have to process an instance of size n^* . At each step in the Markov chain, if X_{m-1} contains a call to process an instance of size 1, we can easily remove that call from the stack when transitioning to X_m , because we assumed that $\mathcal{T}(1) = 0$. Otherwise, we have to perform a call on an instance of size $n_1^{(m-1)}$, which by definition leads to further recursive calls to instances of sizes $h_1(n_1^{(m-1)}), \dots, h_{\ell}(n_1^{(m-1)})$.

Stopping Time and Costs. Let τ be the stopping time of $(X_m)_{m \geq 0}$. Formally, $\tau := \min\{i \mid k_i = 0\}$. We further define $(C_m)_{m \geq 0}$ to model the total cost of execution until reaching X_m :

$$C_m = \begin{cases} 0 & m = 0 \\ C_{m-1} & m > 0 \wedge n_1^{(m-1)} = 1 \\ C_{m-1} + a \left(n_1^{(m-1)} \right) & m > 0 \wedge n_1^{(m-1)} > 1 \end{cases}$$

Following the definitions, it is easy to see that C_τ is the total time cost of running the divide-and-conquer algorithm on an instance of size n^* .

The Exponential Supermartingale of a PRR. Suppose that we are given a function $f : \mathbb{N} \rightarrow [0, \infty)$, such that $f(n) \geq \mathbb{E}[\mathcal{T}(n)]$ for all $n \in \mathbb{N}$. In other words, $f(n)$ is an upper-bound for the expected runtime of our divide-and-conquer algorithm on an instance of size n . Finding such upper-bounds is a well-studied problem and can be automated using approaches such as [Chatterjee et al. 2017]. We define a new stochastic process $(Y_m)_{m \geq 0}$ as follows:

$$Y_m := \sum_{j=1}^{k_m} f(n_j^{(m)}).$$

Moreover, let $\alpha := \alpha(n^*) > 1$ be a constant that depends on the initial input size n^* , and define a new stochastic process $(Z_m)_{m \geq 0}$ defined as $Z_m := \alpha^{C_m + Y_m}$. Note that if Z_m is a supermartingale, then we can obtain a concentration bound for $\mathcal{T}(n^*)$. More formally, we have the following lemma:

LEMMA 5.3. *Let n^* be the size of the initial input instance for the recurrence relation in (2) and $f : \mathbb{N} \rightarrow [0, \infty)$ an upper-bound on the expected value of \mathcal{T} . If for some $\alpha > 1$,*

$$\alpha^{f(n)} \geq \alpha^{a(n)} \cdot \mathbb{E} \left[\alpha^{\sum_{j=1}^{\ell} f(h_j(n))} \right] \quad (3)$$

for all $0 \leq n \leq n^$, then $\Pr[\mathcal{T}(n^*) \geq \kappa] \leq \alpha^{f(n^*) - \kappa}$ for all $\kappa \geq f(n^*)$.*

PROOF. Let (C_m) , (Y_m) , and (Z_m) be defined as above. Equation (3) guarantees that (Z_m) is a supermartingale. Applying Theorem 3.3 to $(C_m + Y_m)$ with $\beta := \alpha$, we obtain $\Pr[\mathcal{T}(n^*) \geq f(n^*) + \kappa'] \leq \alpha^{-\kappa'}$ for all $\kappa' \geq 0$. Substituting $\kappa = f(n^*) + \kappa'$ into the latter inequality leads to the desired concentration bound. \square

5.3 Case Studies

Based on the lemma above, we can now derive concentration bounds for a PRR by synthesizing a suitable α . We now demonstrate the process with a few examples. In the next section, we will show how this process can be automated.

5.3.1 Obtaining Concentration Bounds for QUICKSELECT. Consider the PRR in Example 5.2. We are interested in the tail probability $\Pr[\mathcal{T}(n^*) \geq 12 \cdot n^*]$ and wish to synthesize an upper-bound for this probability. Suppose that the given function f is $f(n) := 5 \cdot n$. In other words, we know that $\mathbb{E}[\mathcal{T}(n)] \leq 5 \cdot n$. We apply Lemma 5.3 to obtain sufficient conditions on α :

$$\forall 1 \leq n \leq n^*, \alpha^{5 \cdot n} \geq \alpha^{n-1} \cdot \frac{1}{n} \cdot \left(\sum_{i=\lceil n/2 \rceil}^{n-1} \alpha^{5 \cdot i} + \sum_{i=\lfloor n/2 \rfloor}^{n-1} \alpha^{5 \cdot i} \right)$$

By simply computing the value of the geometric series, we get:

$$\forall 1 \leq n \leq n^*, \alpha^{4 \cdot n+1} \geq \frac{1}{n} \cdot \left(\frac{\alpha^{5 \cdot n} - \alpha^{5 \cdot \lceil n/2 \rceil} + \alpha^{5 \cdot n} - \alpha^{5 \cdot \lfloor n/2 \rfloor}}{\alpha^5 - 1} \right)$$

since $\alpha^5 - 1 \geq 5 \cdot \ln \alpha$, and $\alpha^{5 \cdot \lceil n/2 \rceil} + \alpha^{5 \cdot \lfloor n/2 \rfloor} \geq 2\alpha^{2.5 \cdot n}$, we strengthen the formula to obtain:

$$\forall 1 \leq n \leq n^*, \alpha^{4 \cdot n} \geq \frac{2}{n} \cdot \left(\frac{\alpha^{5 \cdot n} - \alpha^{2.5 \cdot n}}{5 \cdot \ln \alpha} \right)$$

Let $c := \alpha^n$. We can rewrite the equation above as:

$$\forall 1 \leq n \leq n^*, 5 \cdot c^4 \cdot \ln c \geq 2 \cdot (c^5 - c^{2.5})$$

By basic calculus, we can further prove that $5 \cdot c^4 \cdot \ln c \geq 2 \cdot (c^5 - c^{2.5})$ holds for $c \in [1, 2.74]$. Recall that $c = \alpha^n$. Since for every $\alpha \geq 1$, α^n increases as n increases, our constraint becomes $1 \leq \alpha \wedge \alpha^{n^*} \leq 2.74$, so one possible solution is $\alpha = (2.74)^{1/n^*}$. Plugging this value back into Lemma 5.3, we have $\Pr[\mathcal{T}(n^*) \geq 12 \cdot n^*] \leq (2.74)^{-7} < 0.0009$.

REMARK 2 (COMPARISON WITH [KARP 1994]). *As shown above, our approach is able to synthesize the concentration bound*

$$\Pr[\mathcal{T}(n^*) \geq 12 \cdot n^*] \leq (2.74)^{-7} < 0.0009$$

for the PRR corresponding to QUICKSELECT. In contrast, [Karp 1994] obtains the following concentration bound:

$$\Pr[\mathcal{T}(n^*) \geq 12 \cdot n^*] \leq \left(\frac{3}{4}\right)^8 \approx 0.1001.$$

Hence, our bound is better by a factor of more than 100.

The advantage of our approach is not limited to the specific choice of $12 \cdot n^*$ in the tail probability. See Section 5.5 for concentration bounds for other tail probabilities. We now show how a more general result and a tighter bound can be obtained.

Suppose that we aim to find an upper-bound for the tail probability $\Pr[\mathcal{T}(n^*) \geq r \cdot n^*]$ for an arbitrary constant $r \geq 24$. Let $q > e^2$ be a real number and consider the function $f_q(n) := q \cdot n$. Using a similar calculation as above, defining $c := \alpha^n$, we obtain:

$$\forall 1 \leq n \leq n^*, c^{q/2-1} q \cdot \ln q - 2c^{q/2} + 2 \geq 0$$

Since $q > e^2$, the inequality $c^{q/2-1} \cdot q \cdot \ln q - 2c^{q/2} + 2 \geq 0$ holds for $c \in [1, q]$, so it suffices to find α such that $\alpha^{n^*} \leq q$. We choose $\alpha = q^{1/n^*}$. Plugging this back into Lemma 5.3, leads to:

$$\Pr[\mathcal{T}(n^*) \geq r \cdot n^*] \leq q^{q-r}$$

Specifically, by letting $q = r/\ln r$, we get

$$\Pr[\mathcal{T}(n^*) \geq r \cdot n^*] \leq \left(\frac{r}{\ln r} \right)^{\frac{r}{\ln r} - r}.$$

REMARK 3 (COMPARISON WITH [KARP 1994]). *If we plug $r = 24$ into the general result above, our general approach is able to synthesize the concentration bound*

$$\Pr[\mathcal{T}(n^*) \geq 24 \cdot n^*] \leq \left(\frac{24}{\ln 24} \right)^{\frac{24}{\ln 24} - 24} < 3.612 \times 10^{-15}$$

for the PRR corresponding to QUICKSELECT. In contrast, [Karp 1994] obtains the following concentration bound:

$$\Pr[\mathcal{T}(n^*) \geq 24 \cdot n^*] \leq \left(\frac{3}{4}\right)^{20} \approx 0.00318.$$

Hence, our bound is better by a factor of more than 8.8×10^{11} .

5.3.2 Obtaining Concentration Bounds for QUICKSORT. Consider the PRR in Example 5.1. Our goal is to synthesize an upper-bound for the tail probability $\Pr[\mathcal{T}(n^*) \geq 11 \cdot n^* \cdot \ln n^* + 12 \cdot n^*]$. Given $f(n) := 9 \cdot n \cdot \ln n$, we apply Lemma 5.3 and obtain the following conditions for α :[†]

$$\forall 1 \leq n \leq n^*, \alpha^{9 \cdot n \cdot \ln n} \geq \alpha^{n-1} \cdot \frac{1}{n} \cdot \sum_{i=0}^{n-1} \alpha^{f(i)+f(n-1-i)}$$

For $1 \leq n \leq 8$, we can manually verify that it suffices to set $\alpha > 1$. In the sequel, we assume $n \geq 8$. Note that $(i \cdot \ln i + (n-i-1) \cdot \ln(n-i-1))$ is monotonically decreasing on $[1, \lfloor n/2 \rfloor]$, and monotonically increasing on $[\lceil n/2 \rceil, n]$. We partition the summation above uniformly into eight parts and use the maximum of each part to overapproximate the sum. This leads to the following upper bound for this summation:

$$\begin{aligned} \sum_{i=0}^{n-1} \alpha^{f(i)} &\leq \sum_{j=0}^7 \sum_{i=\lfloor j \cdot n/8 \rfloor}^{\lfloor (j+1) \cdot n/8 \rfloor} \alpha^{f(i)+f(n-1-i)} \\ &\leq \frac{n}{4} \cdot \left(\alpha^{9 \cdot n \cdot \ln n} + \alpha^{9 \cdot (\frac{n}{8} \cdot \ln \frac{n}{8} + \frac{7 \cdot n}{8} \cdot \ln \frac{7 \cdot n}{8})} + \alpha^{9 \cdot (\frac{n}{4} \cdot \ln \frac{n}{4} + \frac{3 \cdot n}{4} \cdot \ln \frac{3 \cdot n}{4})} + \alpha^{9 \cdot (\frac{5 \cdot n}{8} \cdot \ln \frac{5 \cdot n}{8} + \frac{3 \cdot n}{8} \cdot \ln \frac{3 \cdot n}{8})} \right) \end{aligned}$$

Plugging in this overapproximation back into the original inequality, we get:

$$\alpha^{9 \cdot n \cdot \ln n} \geq \alpha^{n-1} \cdot \frac{1}{4} \cdot \left(\alpha^{9 \cdot n \cdot \ln n} + \alpha^{9 \cdot (\frac{n}{8} \cdot \ln \frac{n}{8} + \frac{7 \cdot n}{8} \cdot \ln \frac{7 \cdot n}{8})} + \alpha^{9 \cdot (\frac{n}{4} \cdot \ln \frac{n}{4} + \frac{3 \cdot n}{4} \cdot \ln \frac{3 \cdot n}{4})} + \alpha^{9 \cdot (\frac{5 \cdot n}{8} \cdot \ln \frac{5 \cdot n}{8} + \frac{3 \cdot n}{8} \cdot \ln \frac{3 \cdot n}{8})} \right)$$

for all $8 \leq n \leq n^*$. We define $c := \alpha^n$ to do substitution, and we use the following formula to do strengthening:

$$\begin{aligned} \alpha^{\beta \cdot \ln \beta + (n-\beta) \cdot \ln(n-\beta)} &\leq \alpha^{\beta \cdot \ln n + (n-\beta) \cdot \ln(n-\beta)} \\ &= \alpha^{n \cdot \ln n} \cdot \alpha^{-(n-\beta) \cdot \ln n + (n-\beta) \cdot \ln(n-\beta)} \\ &= \alpha^{n \cdot \ln n + (n-\beta) \cdot \ln \frac{n-\beta}{n}} = c \cdot c^{\frac{(n-\beta)}{n \cdot \ln n} \cdot \ln \frac{n-\beta}{n}} \end{aligned}$$

By defining $\beta = \frac{n}{8}, \frac{n}{4}, \frac{3n}{8}$ respectively, we obtain:

$$\begin{aligned} \forall 8 \leq n \leq n^*, c^{9 \ln n} &\geq \frac{c}{n} \cdot \frac{n}{4} \cdot \left(c^{9 \ln n} + c^{9 \ln n + \frac{63}{8} \cdot \ln \frac{7}{8}} + c^{9 \ln n + \frac{27}{4} \cdot \ln \frac{3}{4}} + c^{9 \ln n + \frac{45}{8} \cdot \ln \frac{5}{8}} \right) \\ \forall 8 \leq n \leq n^*, 4 - \left(c + c^{1 - \frac{63}{8} \cdot \ln \frac{8}{7}} + c^{1 - \frac{27}{4} \cdot \ln \frac{4}{3}} + c^{1 - \frac{45}{8} \cdot \ln \frac{8}{5}} \right) &\geq 0 \end{aligned}$$

Now we study the following function:

$$\psi(c) = 4 - \left(c + c^{1 - \frac{63}{8} \cdot \ln \frac{8}{7}} + c^{1 - \frac{27}{4} \cdot \ln \frac{4}{3}} + c^{1 - \frac{45}{8} \cdot \ln \frac{8}{5}} \right).$$

By basic calculus, we can prove that $\psi(c) \geq 0$ holds on $[1, 2.3]$. Additionally, since for every α , α^n increases as n increases, by plugging $\alpha = 2.3^{1/(n^*)}$ into Lemma 5.3, we obtain:

$$\Pr[\mathcal{T}(n^*) \geq 11 \cdot n^* \cdot \ln n^* + 12 \cdot n^*] \leq (2.3)^{-2 \cdot \ln n^* - 12}.$$

REMARK 4 (COMPARISON WITH [KARP 1994]). As shown above, our approach is able to synthesize a concentration bound that is of the form

$$\Pr[\mathcal{T}(n^*) \geq 11 \cdot n^* \cdot \ln n^* + 12 \cdot n^*] \leq (2.3)^{-2 \cdot \ln n^* - 12}$$

[†]We assume $0 \ln 0 := 0$.

for the PRR corresponding to QUICKSORT. In contrast [Karp 1994] provides the following concentration bound:

$$\Pr[\mathcal{T}(n^*) \geq 10 \cdot n^* \cdot \ln n^*] \leq e^{-4}.$$

Note that the latter bound is a constant, while our bound improves as n^* grows. More concretely, as n^* grows, our bound becomes exponentially better than the one obtained by [Karp 1994].

Just as in the previous case, the advantage of our approach is not limited to bounding $\Pr[\mathcal{T}(n^*) \geq 11 \cdot n^* \cdot \ln n^* + 12 \cdot n^*]$. A similar argument can be applied to obtain similar exponentially-decreasing bounds for $\Pr[\mathcal{T}(n^*) \geq a_1 \cdot n^* \cdot \ln n^* + a_2 \cdot n^*]$ with other values of a_1 and a_2 . Moreover, our bounds improve as a_2 increases. This is in contrast to [Karp 1994] that can only bound $\Pr[\mathcal{T}(n^*) \geq a \cdot n^* \cdot \ln n^*]$. Hence, not only do we beat [Karp 1994]’s bounds numerically, but we also provide a more fine-grained set of concentration bounds.

REMARK 5 (COMPARISON WITH [Tassarotti 2017]). Recall that our approach synthesized the following bound:

$$\Pr[\mathcal{T}(n^*) \geq 11 \cdot n^* \cdot \ln n^* + 12 \cdot n^*] \leq (2.3)^{-2 \cdot \ln n^* - 12},$$

for the PRR corresponding to QUICKSORT, the work of [Tassarotti 2017] improves Karp’s cook-book method and provides the following bound:

$$\Pr[\mathcal{T}(n^*) \geq 11 \cdot n^* \cdot \ln n^* + 12 \cdot n^*] \leq (8/7)^{-(2 - \ln(8/7)) \cdot \ln n^* - 11}$$

Note that our bound beats theirs in both base and exponent, although both bounds are asymptotically equal to $\exp(-\Theta(\ln n^*))$.

REMARK 6 (COMPARISON WITH [McDiarmid and Hayward 1996]). While our approach synthesized the following bound:

$$\Pr[\mathcal{T}(n^*) \geq 11 \cdot n^* \cdot \ln n^* + 12 \cdot n^*] \leq (2.3)^{-2 \cdot \ln n^* - 12},$$

for the PRR corresponding to QUICKSORT, the work of [McDiarmid and Hayward 1996] provides the asymptotically optimal bound of the following form:

$$\Pr[\mathcal{T}(n^*) \geq 11 \cdot n^* \cdot \ln n^* + 12 \cdot n^*] \leq e^{-c \cdot \ln n^* \cdot \ln \ln n^*}$$

where c is a constant. Our bound is comparable but slightly worse than the optimal result in [McDiarmid and Hayward 1996] by only $\ln \ln n^*$ factor. However, their method only works for quick sort, while our method works on a wide-range of algorithms, such as quick select, quick sort and diameter computation. Furthermore, their method is completely manual, while our approach could be automated once the monotonic interval of $f(i) + f(n - 1 - i)$ is obtained, and it is remained as a future work.

5.4 Automated Algorithm

In this section, we consider the problem of automating our PRR analysis. We provide a sound and polynomial time algorithm. Our algorithm is able to automatically synthesize concentration bounds that beat previous methods over several classical benchmarks.

The Setup. In this section, we consider PRRs of the following form:

$$\forall n > 0, \mathcal{T}(n) = a(n) + \mathcal{T}(h(n)) := t \quad \mathcal{T}(1) = 0. \quad (4)$$

in which t is an expression generated by the following grammar:

$$\begin{aligned} t ::= & c \mid n \mid \ln n \mid n \cdot \ln n \mid t + t \mid c \cdot t \\ & \mid \frac{1}{n} \cdot \sum_{j=0}^{n-1} \mathcal{T}(j) \mid \frac{1}{n} \cdot \left(\sum_{i=\lfloor n/2 \rfloor}^{n-1} \mathcal{T}(i) + \sum_{i=\lceil n/2 \rceil}^{n-1} \mathcal{T}(i) \right) \end{aligned}$$

where c is a real constant. We also assume that no sum in t appears with a negative coefficient, and that the coefficient of the most significant non-sum term in t (if it exists) is positive. We focus on the problem of synthesizing upper-bounds for the tail probability $\Pr[\mathcal{T}(n^*) \geq \kappa]$. We also assume that the input to our algorithm contains a function $f : \mathbb{N} \rightarrow [0, \infty)$ that serves as an upper-bound on the expected runtime, i.e. $f(n) \geq \mathbb{E}[\mathcal{T}(n)]$. There are well-known approaches for automatically deriving such functions, e.g. see [Chatterjee et al. 2017]. To enable algorithmic methods, we further assume that κ and $f(n)$ are generated using the following grammar, in which c denotes a real constant:

$$E ::= c \mid E + E \mid c \cdot E \mid n \cdot \ln n \mid \ln n \mid n \quad (5)$$

We also assume that the coefficient of the most significant term in κ and f is positive. The goal of our algorithm is to synthesize an $\alpha > 1$ that satisfies

$$\forall 1 \leq n \leq n^*, \alpha^{f(n)} \geq \alpha^{a(n)} \cdot \mathbb{E} \left[\alpha^{f(h(n))} \right]. \quad (6)$$

Such an α will directly lead to a concentration bound as in Lemma 5.3. Our algorithm relies on the following simple Lemma:

LEMMA 5.4. *For any monotonically increasing function f defined on the interval $[l, r + 1]$, where $l, r \in \mathbb{N}$, we have:*

$$\sum_{i=l}^r f(x) \leq \int_l^{r+1} f(x) dx.$$

Overview of the Algorithm. Our algorithm consists of five steps:

- Step 1.* The algorithm symbolically computes Inequality (6).
- Step 2.* The algorithm replaces every summation in (6) with an over-approximation, hence strengthening the requirements. The algorithm has two ways of obtaining such over-approximations: If the expression inside the sum has an elementary antiderivative that can be computed symbolically, the algorithm applies Lemma 5.4. Otherwise, it finds a bound based on sampling and relying on monotonicity.
- Step 3.* The algorithm introduces a new variable $c := c(\alpha, n)$ and substitutes it into the inequality. It also removes non-significant terms, hence further strengthening the inequality.
- Step 4.* The algorithm uses a calculus technique to obtain a value $c^* > 1$ for c , such that the inequality holds on $[1, c^*]$, but not on $(c^*, +\infty)$. If no such value is found, the algorithm returns the obvious upper-bound 1 for the tail probability.
- Step 5.* The algorithm plugs c^* back into the definition of c and obtains a value for α . Note that this value depends on n^* .

Our Synthesis Algorithm. We now present each step of the algorithm in more detail:

Step 1. Computing Conditions on α . The algorithm creates a variable α and symbolically computes Inequality (6).

Example 5.5. Consider the PRR for QUICKSELECT (Example 5.2) and assume $f(n) := 5 \cdot n$. The algorithm symbolically computes Inequality (6) and obtains the following:

$$\forall 1 \leq n \leq n^*, \alpha^{5 \cdot n} \geq \alpha^{n-1} \cdot \frac{1}{n} \cdot \left(\sum_{i=\lceil n/2 \rceil}^{n-1} \alpha^{5 \cdot i} + \sum_{i=\lfloor n/2 \rfloor}^{n-1} \alpha^{5 \cdot i} \right).$$

Step 2. Over-approximating the Summations. Note that, by design, the expressions inside our summations are monotonically increasing with respect to the summation index. As such, we can apply Lemma 5.4 to obtain an upper-bound for each sum. To do so, the algorithm symbolically computes an antiderivative of the expression inside the sums. Also, note that the antiderivative is always concave, given that the initial expression is increasing. The algorithm uses this concavity property to remove floors and ceilings, hence strengthening the inequality.

However, there are cases where no closed-form or elementary antiderivative can be obtained, e.g. if the expression is $\alpha^{n \cdot \ln n}$. In such cases, the algorithm partitions the summation uniformly into B parts (as in Section 5.3.2) and uses the maximum element of a part to over-approximate each of its elements. Furthermore, to tackle with floors and ceilings in such cases, we would strengthen the inequality by replacing $\lceil n/2 \rceil$ to $\lceil (n-1)/2 \rceil$. The algorithm starts with $B = 2$, and it doubles B to obtain finer over-approximations and repeats the following steps until it succeeds in synthesizing a concentration bound.

Example 5.6. Continuing with the previous example, we have

$$\sum_{i=l}^r \alpha^{5 \cdot i} \leq \int_l^{r+1} \alpha^{5 \cdot x} dx = \frac{\alpha^{5 \cdot (r+1)} - \alpha^{5 \cdot l}}{5 \cdot \ln \alpha}$$

The algorithm applies this over-approximation to obtain the following strengthened inequality:

$$\forall 1 \leq n \leq n^*, \alpha^{5 \cdot n} \geq \alpha^{n-1} \cdot \frac{1}{n} \cdot \frac{\alpha^{5 \cdot n} - \alpha^{5 \cdot \lceil n/2 \rceil} + \alpha^{5 \cdot n} - \alpha^{5 \cdot \lfloor n/2 \rfloor}}{5 \cdot \ln \alpha}$$

The algorithm then further strengthens the inequality by removing the floors and ceilings due to the concavity of $\alpha^{5 \cdot x}$:

$$\forall 1 \leq n \leq n^*, \alpha^{5 \cdot n} \geq \alpha^{n-1} \cdot \frac{2}{n} \cdot \frac{\alpha^{5 \cdot n} - \alpha^{2.5 \cdot n}}{5 \ln \alpha}.$$

Step 3. Substitution and Simplification. Let $g(n)$ be the most significant term in $f(n)$, ignoring constant factors. The algorithm defines a new variable $c = c(\alpha, n) := \alpha^{g(n)} > 1$, and rewrites the inequality based on c . It then moves everything to the LHS, writing the inequality in the form of $e \geq 0$. It also eliminates all fractions by multiplying the inequality by their denominators. This is sound, because by construction, the denominators of all fractions are positive at this point. Then, the algorithm inspects all the terms in e . If a term is of the form $e_1 \cdot c^{e_2}$ in which e_2 contains n as a sub-expression, if $e_1 \cdot e_2 > 0$, then it can be simplified to e_1 , if $e_1 \cdot e_2 < 0$, then we check whether $e_2 \leq 1$ holds, if it holds, it would be simplified into c . This preserves soundness and strengthens the inequality. This preserves soundness and strengthens the inequality. Our algorithm eagerly applies as many such simplifications as possible. Finally, the algorithm divides e by the greatest common divisor of its terms.

Example 5.7. Continuing with the previous example, we have $f(n) = 5 \cdot n$, so the most significant term in $f(n)$ is n . The algorithm therefore defines $c := \alpha^n$, and rewrites the inequality as follows: (Note that $\ln c = n \cdot \ln \alpha$.)

$$\forall 1 \leq n \leq n^*, c^5 \geq c^{1-1/n} \cdot \frac{2 \cdot c^5 - 2 \cdot c^{2.5}}{5 \cdot \ln c}$$

It then moves everything to the LHS, and eliminates the fractions by multiplying their denominators:

$$\forall 1 \leq n \leq n^*, 5 \cdot c^5 \cdot \ln c - 2 \cdot c^{1-1/n} \cdot (c^5 - c^{2.5}) \geq 0$$

Note that $c^{-1/n} < 1$ appears on the LHS with negative coefficient, so its removal would strengthen the inequality. The algorithm simplifies the corresponding term, obtaining the following:

$$\forall 1 \leq n \leq n^*, 5 \cdot c^5 \cdot \ln c - 2 \cdot (c^6 - c^{3.5}) \geq 0$$

The algorithm divides the inequality by $c^{3.5}$, obtaining:

$$\forall 1 \leq n \leq n^*, 5 \cdot c^{1.5} \cdot \ln c - 2 \cdot c^{2.5} + 2 \geq 0$$

Before moving to the next step, we need two simple lemmas:

LEMMA 5.8 (PROOF IN APPENDIX B.1). *After Steps 1–3 above, our inequality is simplified to*

$$\forall 1 \leq n \leq n^*, \psi(c) \geq 0$$

in which $\psi(c)$ is a univariate function of the following form:

$$\psi(c) = \sum_{i \in \mathcal{I}} \mu_i \cdot c^{v_i} \ln^{\xi_i} c \quad (7)$$

where $\mu_i, v_i \in \mathbb{R}$ and $\xi_i \in \{0, 1\}$. Also, note that this form is closed under derivation.

Given that our inequality is now univariate in $c = \alpha^{g(n)}$, we should look for a value $c^* > 1$, such that $\psi(c) \geq 0$ on $[1, c^*]$. Intuitively, if we have such a c^* , then we can let $\alpha := (c^*)^{1/g(n)}$ to solve the problem. Moreover, to find the best possible concentration bound, we would like to find the largest possible c^* . To simplify the matter, we attempt to obtain a c^* such that $\psi(c) \geq 0$ on $[1, c^*]$ and $\psi(c) < 0$ on $(c^*, +\infty)$. Moreover, we say that ψ is *separable* (into non-negative and negative parts) iff such a c^* exists. The following lemma provides sufficient conditions for separability:

LEMMA 5.9 (PROOF IN APPENDIX B.2). *Let $\psi(c)$ be a function of the form (7). If at least one of the following conditions holds, then ψ is separable:*

- (i) ψ is strictly decreasing over $[1, +\infty)$ and $\psi(1) \geq 0$.
- (ii) ψ' is separable and $\psi(1) \geq 0$.
- (iii) ψ/c^a is separable for some constant $a \in \mathbb{R}$.

Step 4. Ensuring Separability and Finding c^* . The algorithm attempts to prove separability of ψ using Lemma 5.9. Rule (iii) of the Lemma is always used to simplify the expression ψ . The algorithm first evaluates $\psi(1)$, ensuring that it is non-negative. Then, it computes the derivative ψ' . If the derivative is negative, then case (i) of Lemma 5.9 is satisfied and ψ is separable. Otherwise, the algorithm tries to recursively prove the separability of ψ' using the same method, hence ensuring case (ii) of the Lemma. If both cases fail, the algorithm has failed to prove the separability of ψ and returns the trivial upper-bound 1. On the other hand, if ψ is proven to be separable, the algorithm obtains c^* by a simple binary search using the fact that for all $c \geq 1$, we have $\psi(c) < 0 \Leftrightarrow c > c^*$.

Example 5.10. Continuing with the previous example, we have

$$\psi(c) = 5 \cdot c^{1.5} \cdot \ln c - 2 \cdot c^{2.5} + 2$$

The algorithm evaluates $\psi(1) = 0$, which is non-negative. Hence, it computes the following derivative:

$$\psi'(c) = 7.5 \cdot c^{0.5} \cdot \ln c + 5 \cdot c^{0.5} - 5 \cdot c^{1.5}$$

Note that $\psi'(c)$ is not always negative for $c \geq 1$. Hence, the algorithm tries to recursively prove that $\psi'(c)$ is separable. It first simplifies ψ' to obtain:

$$\psi_1(c) = 7.5 \cdot \ln c + 5 - 5 \cdot c$$

Now it tries to prove the separability of ψ_1 . It first evaluates $\psi_1(1) = 0$, and then computes the derivative:

$$\psi_1'(c) = \frac{7.5}{c} - 5$$

Another level of recursion shows that $\psi_1'(1) \geq 0$ and ψ_1' is strictly decreasing over $[1, +\infty)$. So, it is separable. Hence, it is proven that ψ is separable, too. The algorithm performs a binary search and obtains $c^* \approx 2.74$.

Step 5. Applying Lemma 5.3. Note that for every α , $c := \alpha^{g(n)}$ increases as n increases. Hence, it suffices to find an α such that $\alpha^{g(1)} > 1$ and $\alpha^{g(n^*)} \leq c^*$. The algorithm calls an off-the-shelf solver to obtain the largest possible α that satisfies these constraints. It then plugs this α into Lemma 5.3 and reports the following concentration bound: $\Pr[\mathcal{T}(n^*) \geq \kappa] \leq \alpha^{f(n^*) - \kappa}$ for all $\kappa \geq f(n^*)$.

Example 5.11. Continuing with previous examples, we had $c = \alpha^n$ and $c^* = 2.74$. So, the algorithm solves the constraints $\alpha > 1$ and $\alpha^{n^*} \leq 2.74$. It is easy to see that $\alpha = (2.74)^{1/n^*}$ is the optimal solution. Hence, the algorithm computes and reports the following concentration bound:

$$\Pr[\mathcal{T}(n^*) \geq \kappa] \leq (2.74)^{5 - \kappa/n^*}$$

for all $\kappa \geq 5 \cdot n^*$. This is equivalent to:

$$\Pr[\mathcal{T}(n^*) \geq \kappa' \cdot n^*] \leq (2.74)^{5 - \kappa'}$$

for all $\kappa' \geq 5$. Note that the bound decreases exponentially as κ' grows.

THEOREM 5.12 (SOUNDNESS). *Given a PRR $\mathcal{T}(n) = a(n) + \mathcal{T}(h(n))$, an expression κ , and an upper-bound function f for the expected runtime of \mathcal{T} , all generated by Grammars (4) and (5), any concentration bound*

$$\forall \kappa \geq f(n^*), \Pr[\mathcal{T}(n^*) \geq \kappa] \leq \alpha^{f(n^*) - \kappa}$$

generated by the algorithm above is a correct bound.

Proof Sketch. While the algorithm strengthens the inequality at some points, it never weakens it. Hence, any concentration bound found by our algorithm is valid, and the algorithm is sound.

THEOREM 5.13. *Assuming fixed bounds on the number of iterations of the binary search, and the approximation parameter B , given a PRR $\mathcal{T}(n) = a(n) + \mathcal{T}(h(n))$, an initial value $n^* \in \mathbb{N}$, an expression κ , and an upper-bound function f for the expected runtime of \mathcal{T} , all generated by Grammars (4) and (5), the algorithm above terminates in polynomial time with respect to the size of input.*

Proof Sketch. Note that each level of recursion in Step 4, i.e. simplification and derivation, strictly reduces the number of terms in the expression that is being studied. Hence, this step performs linearly many symbolic operations. Moreover, Step 5 can find the optimal value of α in $O(|g|)$ operations, where $|g|$ is the length of the expression g (not its value). It is easy to verify that every other step of the algorithm takes polynomial time.

5.5 Prototype Implementation and Experimental Results

Implementation. We implemented the algorithm of Section 5.4 using Python and Mathematica [Wolfram Research 2020]. We used the SymPy package [Meurer et al. 2017] for symbolic differentiation and integration. All results were obtained on a machine with an Intel Core i7-8700 processor (3.2 GHz) and 16 GB of memory, running on Microsoft Windows 10.

Benchmarks. We experimented with PRRs corresponding to 4 classical randomized divide-and-conquer algorithms, namely QUICKSELECT, RANDOMSEARCH, L1DIAMETER, and L2DIAMETER. QUICKSELECT is already described in Section 5.2. In RANDOMSEARCH, the input consists of a sorted array A of n distinct items and a key k . The goal is to find the index of k in the array, or report that it does not appear. The algorithm randomly chooses an index i and compares $A[i]$ with k . If $A[i]$ is larger, it recursively searches the first $i - 1$ locations of the array. If they are equal, it terminates and returns i . If k is larger, it recursively searches the last $n - i$ locations. In L1DIAMETER and L2DIAMETER, the input is a set S of n points in the 3-d space. The goal is to find the diameter of S , or equivalently, to find two points in S that are farthest apart from each other. The only difference between the two problems is the norm that is used for computing the distance (i.e. L1 or L2). Chapter 9 of the classical book [Motwani and Raghavan 1995] provides randomized divide-and-conquer algorithms for these problems.

PRRs. The PRRs used in our experiments are shown in the table below.

Randomized Algorithm	Probabilistic Recurrence Relation
QUICKSELECT	$\mathcal{T}(n) = n - 1 + \frac{1}{n} \left(\sum_{i=\lceil n/2 \rceil}^{n-1} \mathcal{T}(i) + \sum_{i=\lfloor n/2 \rfloor}^{n-1} \mathcal{T}(i) \right)$
RANDOMSEARCH	$\mathcal{T}(n) = 1 + \frac{1}{n} \left(\sum_{i=\lceil n/2 \rceil}^{n-1} \mathcal{T}(i) + \sum_{i=\lfloor n/2 \rfloor}^{n-1} \mathcal{T}(i) \right)$
L1DIAMETER	$\mathcal{T}(n) = n + \frac{1}{n} \left(\sum_{i=0}^{n-1} \mathcal{T}(i) \right)$
L2DIAMETER	$\mathcal{T}(n) = n \ln n + \frac{1}{n} \left(\sum_{i=0}^{n-1} \mathcal{T}(i) \right)$

Results. Our experimental results are shown in Tables 2 and 3. For each recurrence relation, we consider several different tail probabilities, and manually compute and report the concentration bound obtained by the classical method of [Karp 1994]. We then provide results of our algorithm using various different functions f . Recall that $f(n)$ is an upper-bound for $\mathbb{E}[\mathcal{T}(n)]$. In cases where our algorithm was unable to find antiderivatives, we also report the parameter B , i.e. the number of blocks used in over-approximating the summations.

Discussion. As shown in Tables 2 and 3, our algorithm is very efficient and can handle the benchmarks in a few seconds. Moreover, it obtains concentration bounds that are consistently tighter than those of [Karp 1994], often by one or more orders of magnitude (see the ratio columns). It is also noteworthy that, for the RANDOMSEARCH benchmark, our algorithm obtains concentration bounds that decrease as n^* goes up, whereas [Karp 1994] only provides constant bounds. In this case, the ratio becomes arbitrarily large as n^* grows.

6 RELATED WORKS

Previous results on concentration bounds for probabilistic programs. Concentration bounds for probabilistic programs were first considered by [Monniaux 2001] where a basic approach for obtaining exponentially-decreasing concentration bounds through abstract interpretation and truncation of the sampling intervals is presented. The work of [Chakarov and Sankaranarayanan

Table 2. Experimental results over the PRRs corresponding to QUICKSELECT and RANDOMSEARCH

Recurrence	Tail Probability	Karp's bound	f(n)	Our bound	Our runtime (s)	Ratio of the bounds \approx	B
QUICKSELECT	$\Pr[\mathcal{T}(n^*) \geq 17n^*]$	$(\frac{3}{4})^{13} \approx 0.024$	$6n$	$4.36 \cdot 10^{-8}$	3.27	$5 \cdot 10^5$	/
			$7n$	$6.11 \cdot 10^{-9}$	3.24	$3.63 \cdot 10^6$	
			$10n$	$1.86 \cdot 10^{-8}$	3.34	$1.29 \cdot 10^6$	
	$\Pr[\mathcal{T}(n^*) \geq 15n^*]$	$(\frac{3}{4})^{11} \approx 0.043$	$9n$	$6.89 \cdot 10^{-7}$	2.32	$6.24 \cdot 10^4$	
			$12.5n$	0.002	2.18	21.5	
			$13n$	0.005	2.09	8.6	
	$\Pr[\mathcal{T}(n^*) \geq 11n^*]$	$(\frac{3}{4})^7 \approx 0.021$	$5n$	0.0024	2.25	8.75	
			$6n$	0.0021	2.10	10	
			$8n$	0.0016	2.15	13.125	
	$\Pr[\mathcal{T}(n^*) \geq 8n^*]$	$(\frac{3}{4})^4 \approx 0.317$	$5.5n$	0.039	2.42	8.12	
			$6n$	0.046	1.82	6.89	
			$7n$	0.151	2.28	2.10	
	$\Pr[\mathcal{T}(n^*) \geq 6n^*]$	$(\frac{3}{4})^2 = 0.5625$	$4.5n$	0.406	2.78	1.39	
			$5n$	0.365	2.67	1.54	
			$5.2n$	0.402	3.45	1.39	
RANDOMSEARCH	$\Pr[\mathcal{T}(n^*) \geq 11 \ln n^*]$	$(\frac{3}{4})^{11 - \frac{1}{\ln \frac{3}{4}}} \approx 0.12$	$5 \ln n$	$(n^*)^{-8.24}$	3.12	$+\infty$ as $n^* \rightarrow \infty$	/
			$7 \ln n$	$(n^*)^{-8.11}$	3.07		
			$9 \ln n$	$(n^*)^{-6.75}$	3.20		
	$\Pr[\mathcal{T}(n^*) \geq 10 \ln n^*]$	$(\frac{3}{4})^{10 - \frac{1}{\ln \frac{3}{4}}} \approx 0.154$	$7 \ln n$	$(n^*)^{-6.08}$	2.28		
			$8.5 \ln n$	$(n^*)^{-3.52}$	2.90		
			$9.5 \ln n$	$(n^*)^{-1.26}$	2.37		
	$\Pr[\mathcal{T}(n^*) \geq 8 \ln n^*]$	$(\frac{3}{4})^{8 - \frac{1}{\ln \frac{3}{4}}} \approx 0.273$	$5.5 \ln n$	$(n^*)^{-3.94}$	2.50		
			$6 \ln n$	$(n^*)^{-3.49}$	2.37		
			$6.5 \ln n$	$(n^*)^{-2.84}$	2.26		
	$\Pr[\mathcal{T}(n^*) \geq 7 \ln n^*]$	$(\frac{3}{4})^{7 - \frac{1}{\ln \frac{3}{4}}} \approx 0.363$	$4.5 \ln n$	$(n^*)^{-2.80}$	2.29		
			$5.5 \ln n$	$(n^*)^{-2.36}$	2.47		
			$6 \ln n$	$(n^*)^{-1.74}$	2.40		
	$\Pr[\mathcal{T}(n^*) \geq 5 \ln n^*]$	$(\frac{3}{4})^{5 - \frac{1}{\ln \frac{3}{4}}} \approx 0.645$	$3.7 \ln n$	$(n^*)^{-0.68}$	2.37		
			$4 \ln n$	$(n^*)^{-0.78}$	2.61		
			$4.5 \ln n$	$(n^*)^{-0.56}$	2.29		

2013] used Azuma's inequality to derive exponentially-decreasing concentration results for values of program variables in a probabilistic program. For termination time of probabilistic programs exponentially-decreasing concentration bounds for special classes of probabilistic programs using Azuma and Hoeffding inequalities were established in [Chatterjee et al. 2018b]. Recently, for several cases where the Azuma and Hoeffding inequalities are not applicable, a reciprocal concentration bound using Markov's inequality was presented in [Fu and Chatterjee 2019], which was then extended to higher moments in [Kura et al. 2019; Wang et al. 2020]. For a detailed survey of the current methods of concentration bounds for probabilistic programs see [Sankaranarayanan 2020].

Previous results on concentration bounds for probabilistic recurrences. Concentration-bound analyses for probabilistic recurrence relations were first considered in the classical work of [Karp 1994], where cookbook methods (similar to the master theorem for worst-case analysis of recurrences) were obtained for a large class of probabilistic recurrence relations. A variant of the results of Karp that weakened several conditions but obtained comparable bounds was presented in [Chaudhuri and Dubhashi 1997]. The optimal concentration bound for the QUICKSORT algorithm was presented in [McDiarmid and Hayward 1996]. Recently, concentration bounds for probabilistic

Table 3. Experimental results over the PRRs corresponding to L1DIAMETER and L2DIAMETER

Recurrence	Tail Probability	Karp's bound	f(n)	Our bound	Our runtime (s)	Ratio of the bounds \approx	B
L1DIAMETER	$\Pr[\mathcal{T}(n^*) \geq 13n^*]$	$(\frac{1}{2})^{11} \approx 4.89 \cdot 10^{-4}$	$4.3n$	$2.33 \cdot 10^{-9}$	3.76	$2.09 \cdot 10^5$	/
			$5n$	$1.47 \cdot 10^{-9}$	2.97	$3.32 \cdot 10^5$	
			$5.2n$	$1.48 \cdot 10^{-9}$	3.34	$3.34 \cdot 10^5$	
	$\Pr[\mathcal{T}(n^*) \geq 11n^*]$	$(\frac{1}{2})^9 = 0.002$	$2.5n$	$1.891 \cdot 10^{-4}$	2.09	10.58	
			$6n$	$1.317 \cdot 10^{-6}$	1.80	1518.61	
			$7n$	$1.976 \cdot 10^{-5}$	1.82	101.21	
	$\Pr[\mathcal{T}(n^*) \geq 9n^*]$	$(\frac{1}{2})^7 = 0.008$	$2.5n$	0.002	2.33	4	
			$5.5n$	$7.957 \cdot 10^{-5}$	2.27	25.14	
			$6n$	$2.963 \cdot 10^{-4}$	2.00	6.75	
	$\Pr[\mathcal{T}(n^*) \geq 7n^*]$	$(\frac{1}{2})^5 = 0.032$	$3.5n$	0.002	2.07	16	
			$5n$	0.007	2.07	4.571	
			$5.5n$	0.018	2.50	1.778	
	$\Pr[\mathcal{T}(n^*) \geq 5n^*]$	$(\frac{1}{2})^3 = 0.125$	$2.5n$	0.081	2.68	1.54	
			$3n$	0.046	2.78	2.717	
			$4n$	0.117	2.56	1.068	
L2DIAMETER	$\Pr[\mathcal{T}(n^*) \geq 20n^* \ln n^*]$	$(\frac{1}{2})^{18} \approx 3.81 \cdot 10^{-6}$	$3.5n \ln n$	$2.07 \cdot 10^{-6}$	4.90	1.84	2
			$5n \ln n$	$5.51 \cdot 10^{-10}$	15.42	6914.7	4
	$\Pr[\mathcal{T}(n^*) \geq 15n^* \ln n^*]$	$(\frac{1}{2})^{13} \approx 1.23 \cdot 10^{-4}$	$5n \ln n$	$6.715 \cdot 10^{-7}$	12.41	567.38	4
			$7n \ln n$	$1.41 \cdot 10^{-5}$	14.21	27.02	4
	$\Pr[\mathcal{T}(n^*) \geq 13.5n^* \ln n^*]$	$(\frac{1}{2})^{11.5} \approx 3.45 \cdot 10^{-4}$	$2.5n \ln n$	$7.22 \cdot 10^{-5}$	7.51	5.27	2
			$5n \ln n$	$5.67 \cdot 10^{-6}$	14.75	67.19	4
	$\Pr[\mathcal{T}(n^*) \geq 9n^* \ln n^*]$	$(\frac{1}{2})^7 = 0.008$	$2.5n \ln n$	0.004	5.64	2	2
			$4.5n \ln n$	0.001	12.44	8	4
	$\Pr[\mathcal{T}(n^*) \geq 8n^* \ln n^*]$	$(\frac{1}{2})^6 \approx 0.016$	$2.5n \ln n$	0.009	6.14	1.79	2
			$4.5n \ln n$	0.007	15.13	2.29	4

recurrence relations were mechanized in a theorem prover [Tassarotti and Harper 2018], and extended to parallel settings [Tassarotti 2017].

Comparison with previous approaches on probabilistic programs. Compared with previous results on probabilistic programs, our approach considers synthesis of exponential supermartingales. As compared to [Monniaux 2001], our approach is based on the well-studied theory of martingales for stochastic processes. In comparison to previous martingale-based approaches, we either achieve asymptotically better bounds (e.g. we achieve exponentially-decreasing bounds as compared to polynomially-decreasing bounds of [Fu and Chatterjee 2019; Kura et al. 2019; Wang et al. 2020]) or substantially improve the bounds (e.g. in comparison with [Chatterjee et al. 2018b] (see our experimental results in Section 4.3). Moreover, all previous results, such as [Chakarov and Sankaranarayanan 2013; Chatterjee et al. 2018b], that achieve exponentially-decreasing bounds require bounded-difference, i.e. the stepwise difference in a supermartingale needs to be globally bounded to apply Azuma or Hoeffding inequalities. In contrast, our results can apply to stochastic processes that are not necessarily difference-bounded.

Comparison with previous approaches on probabilistic recurrences. Compared with previous results on probabilistic recurrence relations, our result is based on the idea of exponential supermartingales and related automation techniques. It can derive much better concentration bounds over the classical approach of [Karp 1994] (See Remarks 2,3, and 4 in Section 5.3), and beats the manual approach of [Tassarotti 2017] in constants (See Remark 5). Moreover, our approach also derives a tail bound for QUICKSORT that is comparable to the optimal bound proposed in [McDiarmid

and Hayward 1996] (see Remark 6 in Section 5.3). In addition, the result of [Karp 1994] requires the key condition $\sum_i \mathbb{E}(T(h_i(n))) \leq \mathbb{E}(T(n))$ to handle recurrences with multiple procedure calls. Whether this condition can be relaxed is raised as an important open problem in [Dubhashi and Panconesi 2009; Karp 1994]. To address this issue, the approach of [Tassarotti 2017] proposed a method that does not require this restriction. Our method also does not need this restriction, hence could be viewed as a new way to resolve this problem.

Key conceptual difference. Finally, our general approach of exponential supermartingales has a key difference with respect to previous approaches. The main conceptual difference is that our approach examines the detailed probability distribution (as moment generating function). In comparison, previous approaches (e.g. those based Azuma and Hoeffding inequalities, or results of [Karp 1994]) only consider the expectation, the variance, or the range of related random variables. This is the key conceptual reason that our approach can derive tighter bounds.

7 CONCLUSION AND FUTURE WORK

In this work, we presented a new approach to derive tighter concentration bounds for probabilistic programs and probabilistic recurrence relations. We showed that our new approach can derive tighter concentration bounds than the classical methods of applying Azuma's inequality for probabilistic programs and the classical approaches for probabilistic recurrences, even for basic randomized algorithms such as QUICKSELECT and randomized diameter computation. On QUICKSORT, we beat the approach of [Karp 1994] and [Tassarotti 2017], and derive a bound comparable to the optimal bound in [McDiarmid and Hayward 1996].

There are several interesting directions for future work. First, while we consider classical probability distributions such as uniform and Bernoulli for algorithmic aspects, extending the algorithms to handle various other distributions is an interesting direction. Second, whether our technique can be used for the relaxation of the key condition $\sum_i \mathbb{E}(T(h_i(n))) \leq \mathbb{E}(T(n))$ in the approach of [Karp 1994] is another interesting problem. Finally, whether our approach can be directly applied to analyze randomized algorithms, rather than their corresponding probabilistic recurrences, is another direction of future work.

8 ACKNOWLEDGEMENTS

We sincerely thank Prof. Joseph Tassarotti for pointing out a miscalculation in Remark 6.

REFERENCES

- Sheshansh Agrawal, Krishnendu Chatterjee, and Petr Novotný. 2018. Lexicographic ranking supermartingales: an efficient approach to termination of probabilistic programs. In *POPL*. 34:1–34:32.
- Kazuoki Azuma. 1967. Weighted sums of certain dependent random variables. *Tohoku Mathematical Journal* 19, 3 (1967), 357–367.
- Christel Baier and Joost-Pieter Katoen. 2008. *Principles of model checking*. MIT Press.
- Gilles Barthe, Thomas Espitau, Benjamin Grégoire, Justin Hsu, and Pierre-Yves Strub. 2018. Proving expected sensitivity of probabilistic programs. In *POPL*. 57:1–57:29.
- Gilles Barthe, Benjamin Grégoire, Justin Hsu, and Pierre-Yves Strub. 2017. Coupling proofs are probabilistic product programs. In *POPL*. 161–174.
- Gilles Barthe, Joost-Pieter Katoen, and Alexandra Silva (Eds.). 2020. *Foundations of Probabilistic Programming*. Springer.
- Aleksandar Chakarov and Sriram Sankaranarayanan. 2013. Probabilistic Program Analysis with Martingales. In *CAV 2013*. 511–526.
- Krishnendu Chatterjee, Hongfei Fu, and Amir Kafshdar Goharshady. 2016. Termination Analysis of Probabilistic Programs Through Positivstellensatz's. In *CAV*. 3–22.
- Krishnendu Chatterjee, Hongfei Fu, Amir Kafshdar Goharshady, and Nastaran Okati. 2018a. Computational Approaches for Stochastic Shortest Path on Succinct MDPs. In *IJCAI 2018*. 4700–4707.

- Krishnendu Chatterjee, Hongfei Fu, and Aniket Murhekar. 2017. Automated Recurrence Analysis for Almost-Linear Expected-Runtime Bounds. In *CAV*. 118–139.
- Krishnendu Chatterjee, Hongfei Fu, Petr Novotný, and Rouzbeh Hasheminezhad. 2018b. Algorithmic Analysis of Qualitative and Quantitative Termination Problems for Affine Probabilistic Programs. *TOPLAS* 40, 2 (2018), 7:1–7:45.
- Shiva Chaudhuri and Devdatt P. Dubhashi. 1997. Probabilistic Recurrence Relations Revisited. *Theoretical Computer Science* 181, 1 (1997), 45–56.
- Marco Cusumano-Towner, Benjamin Bichsel, Timon Gehr, Martin T. Vechev, and Vikash K. Mansinghka. 2018. Incremental inference for probabilistic programs. In *PLDI*. 571–585.
- Fredrik Dahlqvist and Dexter Kozen. 2020. Semantics of higher-order probabilistic programs with conditioning. In *POPL*. 57:1–57:29.
- Devdatt P. Dubhashi and Alessandro Panconesi. 2009. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press.
- R. Durrett. 1996. *Probability: Theory and Examples (Second Edition)*. Duxbury Press.
- Javier Esparza, Andreas Gaiser, and Stefan Kiefer. 2012. Proving Termination of Probabilistic Programs Using Patterns. In *CAV*. 123–138.
- Luis María Ferrer Fioriti and Holger Hermanns. 2015. Probabilistic Termination: Soundness, Completeness, and Compositionality. In *POPL*. 489–501.
- Nate Foster, Dexter Kozen, Konstantinos Mamouras, Mark Reitblatt, and Alexandra Silva. 2016. Probabilistic NetKAT. In *ESOP*. 282–309.
- Hongfei Fu and Krishnendu Chatterjee. 2019. Termination of Nondeterministic Probabilistic Programs. In *VMCAI*. 468–490.
- Marcel Hark, Benjamin Lucien Kaminski, Jürgen Giesl, and Joost-Pieter Katoen. 2020. Aiming low is harder: induction for lower bounds in probabilistic program verification. In *POPL*. 37:1–37:28.
- Charles Antony Richard Hoare. 1961a. Algorithm 64: quicksort. *Commun. ACM* 4, 7 (1961), 321.
- Charles Antony Richard Hoare. 1961b. Algorithm 65: find. *Commun. ACM* 4, 7 (1961), 321–322.
- H. Howard. 1960. *Dynamic Programming and Markov Processes*. MIT Press.
- L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. 1998. Planning and acting in partially observable stochastic domains. *Artificial intelligence* 101, 1 (1998), 99–134.
- L. P. Kaelbling, M. L. Littman, and A. W. Moore. 1996. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research* 4 (1996), 237–285.
- Benjamin Lucien Kaminski, Joost-Pieter Katoen, Christoph Matheja, and Federico Olmedo. 2016. Weakest Precondition Reasoning for Expected Run-Times of Probabilistic Programs. In *ESOP*. 364–389.
- Richard M. Karp. 1994. Probabilistic Recurrence Relations. *Journal of the ACM* 41, 6 (1994), 1136–1150.
- J.G. Kemeny, J.L. Snell, and A.W. Knapp. 1966. *Denumerable Markov Chains*. Van Nostrand Company.
- Jon M. Kleinberg and Éva Tardos. 2006. *Algorithm design*. Addison-Wesley.
- Satoshi Kura, Natsuki Urabe, and Ichiro Hasuo. 2019. Tail Probabilities for Randomized Program Runtimes via Martingales for Higher Moments. In *TACAS*.
- Marta Z. Kwiatkowska, Gethin Norman, and David Parker. 2011. PRISM 4.0: Verification of Probabilistic Real-Time Systems. In *CAV*. 585–591.
- Colin McDiarmid and Ryan Hayward. 1996. Large Deviations for Quicksort. *Journal of Algorithms* 21, 3 (1996), 476–507.
- Aaron Meurer, Christopher P Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B Kirpichev, Matthew Rocklin, Amit Kumar, Sergiu Ivanov, Jason K Moore, Sartaj Singh, et al. 2017. SymPy: symbolic computing in Python. *PeerJ Computer Science* 3 (2017).
- S.P. Meyn and R.L. Tweedie. 1993. *Markov chains and stochastic stability*. Springer.
- David Monniaux. 2001. An Abstract Analysis of the Probabilistic Termination of Programs. In *SAS*. 111–126.
- Rajeev Motwani and Prabhakar Raghavan. 1995. *Randomized Algorithms*. Cambridge University Press.
- Van Chan Ngo, Quentin Carbonneaux, and Jan Hoffmann. 2018. Bounded expectations: resource analysis for probabilistic programs. In *PLDI*. 496–512.
- Federico Olmedo, Benjamin Lucien Kaminski, Joost-Pieter Katoen, and Christoph Matheja. 2016. Reasoning about Recursive Probabilistic Programs. In *LICS*, Martin Grohe, Eric Koskinen, and Natarajan Shankar (Eds.). 672–681.
- A. Paz. 1971. *Introduction to probabilistic automata (Computer science and applied mathematics)*. Academic Press.
- M.O. Rabin. 1963. Probabilistic automata. *Information and Control* 6 (1963), 230–245.
- Sriram Sankaranarayanan. 2020. Quantitative Analysis of Programs with Probabilities and Concentration of Measure Inequalities. In *Foundations of Probabilistic Programming*, Gilles Barthe, Joost-Pieter Katoen, and Alexandra Silva (Eds.). <https://www.cs.colorado.edu/~srirams/papers/concMeasureSpringerChapter.PDF>
- Sriram Sankaranarayanan, Aleksandar Chakarov, and Sumit Gulwani. 2013. Static analysis for probabilistic programs: inferring whole program properties from finitely many paths. In *PLDI*. 447–458.

- Joseph Tassarotti. 2017. Probabilistic Recurrence Relations for Work and Span of Parallel Algorithms. *CoRR* abs/1704.02061 (2017). <http://arxiv.org/abs/1704.02061>
- Joseph Tassarotti and Robert Harper. 2018. Verified Tail Bounds for Randomized Programs. In *ITP*. 560–578.
- Di Wang, Jan Hoffmann, and Thomas W. Reps. 2020. Tail Bound Analysis for Probabilistic Programs via Central Moments. *CoRR* abs/2001.10150 (2020). <https://arxiv.org/abs/2001.10150>
- David Williams. 1991a. *Probability with Martingales*. Cambridge University Press.
- David Williams. 1991b. *Probability with martingales*. Cambridge university press.
- Wolfram Research. 2020. Mathematica, Version 12.1. <https://www.wolfram.com/mathematica>

A DETAILED EXPERIMENTAL RESULTS FOR SECTION 4.3

In this section, there are some figures of related example in section 4.3 in Figure 1. The numbers on the x-axis are the terminating time n varying from $6X_0$ to $22X_0$.

B DETAILED PROOFS FOR SECTION 5

B.1 Proof of Lemma 5.8

It is easy to see that this form is closed under addition, multiplication and derivative. We first prove the following lemma:

LEMMA B.1. *For $0 \leq h(n) \leq f(n)$ generated by c , suppose $g(n)$ is the most significant term (ignoring coefficients), which means $f(n) := q \cdot g(n) + o(g(n))$. Set $c := \alpha^{g(n)}$, then for every $u \in \mathbb{R}$, $u \cdot \alpha^{h(n)}$ could be simplified under Step 3's strategy into a univariate function on c as in Lemma 5.8.*

PROOF. Suppose $h(n) := \sum_i \omega_i h_i(n)$, where $h_i(n) \in \{n, \ln n, n \ln n\}$. Then we have:

$$\begin{aligned} u \cdot \alpha^{h(n)} &= u \cdot c^{\frac{h(n)}{g(n)}} \\ &= u \cdot \prod_i c^{\omega_i \frac{h_i(n)}{g(n)}} \end{aligned}$$

For every i , if $h_i(n) = g(n)$, then this term would become c^{ω_i} , otherwise $\frac{h_i(n)}{g(n)} \leq 1$ since $h(n) \leq f(n)$. Then if $\omega_i \cdot u > 0$, it would be simplified into 1, otherwise it would be simplified into c . Since the form is closed under multiplication, we conclude that $u \cdot \alpha^{h(n)}$ could be simplified into the form of Lemma 5.8. \square

We now return to proving Lemma 5.8. By design, our PRR always has the following form:

$$\mathcal{T}(n) = a(n) + \frac{\gamma}{n} \cdot \left(\sum_{i=\lceil n/2 \rceil}^{n-1} \mathcal{T}(i) + \sum_{i=\lfloor n/2 \rfloor}^{n-1} \mathcal{T}(i) \right) + \frac{1-\gamma}{n} \cdot \sum_{i=0}^{n-1} \mathcal{T}(i)$$

where $0 \leq \gamma \leq 1$. We consider several cases:

Case (i). $f(n) := q \ln n$ and $q > 0$. Recall that $f(n)$ is a over-approximation of $\mathbb{E}[\mathcal{T}(n)]$, $a(n)$ would be $k_1 \ln n + k_2$. First note that $k_1 = 0$, since if $k_1 > 0$, then $\mathbb{E}[\mathcal{T}(n)] = \Omega(\ln^2 n) > f(n)$. Since $k_1 = 0$, we have $k_2 \geq 0$. Hence, the conditions for α would be:

$$\alpha^{q \cdot \ln n} \geq \alpha^{k_2} \cdot \left(\frac{\gamma}{n} \cdot \left(\sum_{i=\lceil n/2 \rceil}^{n-1} \alpha^{q \cdot \ln i} + \sum_{i=\lfloor n/2 \rfloor}^{n-1} \alpha^{q \cdot \ln i} \right) + \frac{1-\gamma}{n} \cdot \sum_{i=0}^{n-1} \alpha^{q \cdot \ln i} \right)$$

By integration and concavity, we have:

$$\alpha^{q \cdot \ln n} \geq \alpha^{k_2} \cdot \left(\frac{2\gamma}{n} \cdot \left(\frac{n\alpha^{q \ln n} - (n/2)\alpha^{q \ln(n/2)}}{q \ln \alpha + 1} \right) + \frac{1-\gamma}{n} \cdot \frac{n\alpha^{q \ln n}}{q \ln \alpha + 1} \right)$$

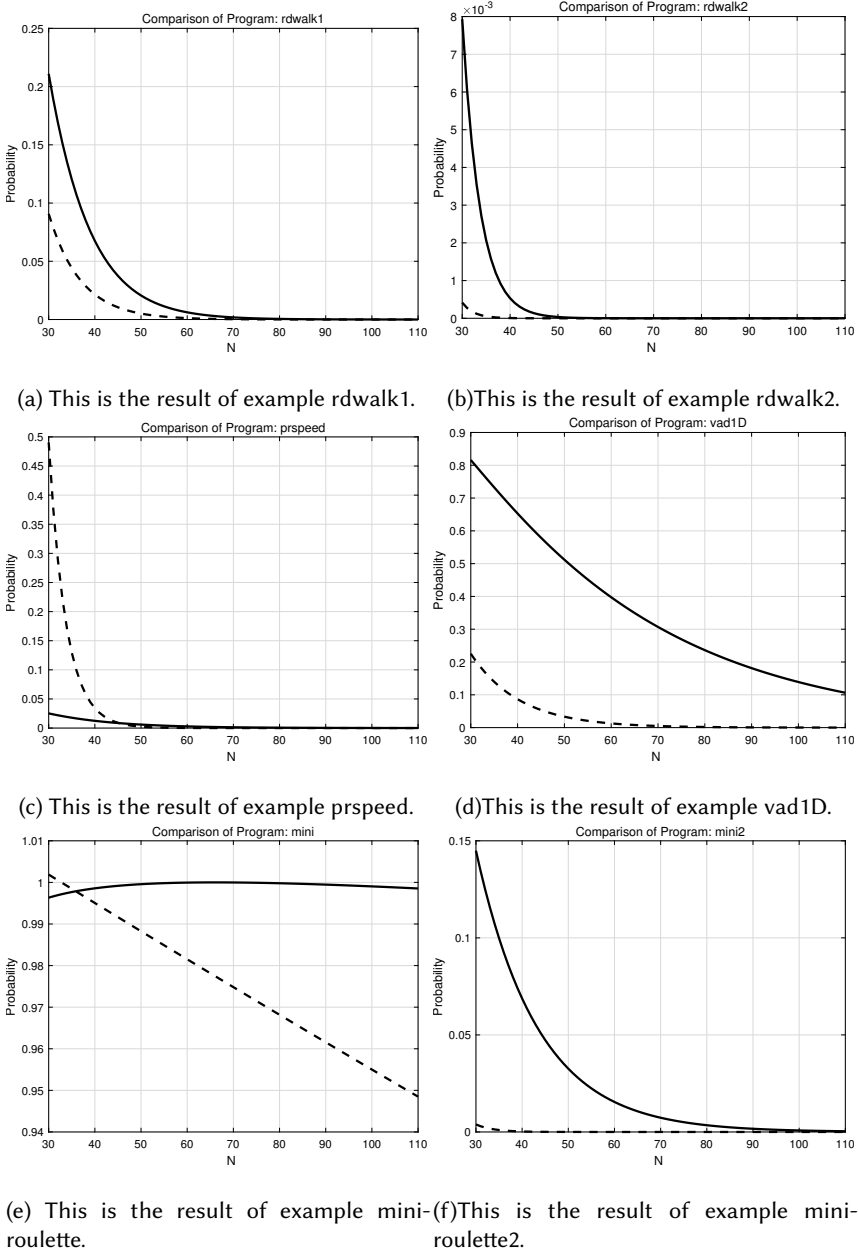


Fig. 1. They are specific figures of experiment results. In these figures, the solid line represents the Hoeffding Bound and the dotted line represents the bound from our approach.

We define $c := \alpha^{\ln n}$, we have:

$$c^q \geq \alpha^{k_2} \cdot \left(\frac{2\gamma}{n} \cdot \left(\frac{nc^q - (n/2)(\frac{c^q}{\alpha^{q \ln 2}})}{q \ln \alpha + 1} \right) + \frac{1 - \gamma}{n} \cdot \frac{nc^q}{q \ln \alpha + 1} \right)$$

By moving everything to left and eliminating the fraction, we get:

$$nc^q(1 + q \ln \alpha) - \alpha^{k_2} \cdot 2\gamma \cdot \left(nc^q - (n/2) \cdot \frac{c^q}{\alpha^{q \ln 2}} \right) + (1 - \gamma) \cdot (n \cdot c^q) \geq 0$$

By removing common parts n and c^q , we derive:

$$(1 + q \ln \alpha) - \alpha^{k_2} \cdot 2\gamma \cdot \left(1 - (1/2) \cdot \frac{1}{\alpha^{q \ln 2}} \right) + (1 - \gamma) \geq 0$$

Since k_2, γ, q are constants, this is a univariate inequality for α in the form of Lemma 5.8.

Case (ii). $f(n) := qn, q > 0$. Recall that $f(n)$ is a over-approximation of $\mathbb{E}[\mathcal{T}(n)]$. $a(n)$ would be $k_1 n + k_2 \ln n + k_3$, where $0 < k_1 \leq q, k_2, k_3 \in \mathbb{R}$. Conditions for α would be:

$$\alpha^{q \cdot n} \geq \alpha^{k_1 \cdot n + k_2 \cdot \ln n + k_3} \cdot \left(\frac{\gamma}{n} \cdot \left(\sum_{i=\lceil n/2 \rceil}^{n-1} \alpha^{q \cdot i} + \sum_{i=\lfloor n/2 \rfloor}^{n-1} \alpha^{q \cdot i} \right) + \frac{1 - \gamma}{n} \cdot \sum_{i=0}^{n-1} \alpha^{q \cdot i} \right)$$

By integration and concavity, we have:

$$\alpha^{q \cdot n} \geq \alpha^{k_1 \cdot n + k_2 \cdot \ln n + k_3} \cdot \left(\frac{2\gamma}{n} \cdot \left(\frac{\alpha^{q \cdot n} - \alpha^{q \cdot (n/2)}}{q \ln \alpha} \right) + \frac{1 - \gamma}{n} \cdot \frac{\alpha^{q \cdot n} - 1}{q \ln \alpha} \right)$$

By moving everything to left and eliminating the fraction, we get:

$$n \cdot q \cdot \ln \alpha \cdot \alpha^{q \cdot n} - \alpha^{k_1 \cdot n + k_2 \cdot \ln n + k_3} \cdot \left(2\gamma \cdot \left(\alpha^{q \cdot n} - \alpha^{q \cdot (n/2)} \right) + (1 - \gamma) \cdot (\alpha^{q \cdot n} - 1) \right) \geq 0$$

We define $c := \alpha^n$. By Lemma B.1, $\alpha^{k_1 \cdot n + k_2 \cdot \ln n + k_3}$, and since $n \ln \alpha = \ln c$ could be simplified into the form we want, thus the whole formula could be simplified into the form we want.

Case (iii). Otherwise, by design, $\alpha^{f(n)}$ will have no elementary antiderivative, in this case we would partition the summation uniformly into B parts, and use the maximum element of a part to over-approximate each of its elements. In this case, conditions for α would be:

$$\alpha^{f(n)} \geq \alpha^{a(n)} \cdot \left(\frac{\gamma}{n} \cdot \left(2 \sum_{i=\lceil (n-1)/2 \rceil}^{n-1} \alpha^{f(i)} \right) + \frac{1 - \gamma}{n} \cdot \sum_{i=0}^{n-1} \alpha^{f(i)} \right)$$

By uniformly dividing into B parts and using the maximum element of a part to over-approximate each of its elements, we derive:

$$\alpha^{f(n)} \geq \alpha^{a(n)} \cdot \left(\frac{\gamma}{n} \cdot \left(\frac{2n}{B} \cdot \sum_{i=1}^B \alpha^{f\left(\frac{n(B+i)}{2B}\right)} \right) + \frac{1 - \gamma}{n} \cdot \frac{n}{B} \cdot \sum_{i=1}^B \alpha^{f\left(\frac{in}{B}\right)} \right)$$

By moving everything to the left, and removing the denominators, we obtain:

$$B \cdot n \cdot \alpha^{f(n)} - \alpha^{a(n)} \cdot \left(2 \cdot \gamma \cdot n \cdot \sum_{i=1}^B \alpha^{f\left(\frac{n(B+i)}{2B}\right)} + n \cdot (1 - \gamma) \cdot \sum_{i=1}^B \alpha^{f\left(\frac{in}{B}\right)} \right) \geq 0$$

We remove the common term n , and let $c := \alpha^{g(n)}$, where $g(n)$ is the most significant term in $f(n)$ (ignoring coefficients). By Lemma B.1, the whole formula could be simplified into the form we want.

B.2 Proof of Lemma 5.9

Proof for Case (i). Define $\psi(\infty) := \lim_{t \rightarrow \infty} \psi(t) \in \mathbb{R} \cup \{-\infty\}$. Set $c^* := \sup\{x | x \geq 1 \wedge \psi(x) \geq 0\}$, then by $\forall x > c^*, \psi(x) < 0$ and continuity of ψ , we have $\psi(c^*) \geq 0$. Since ψ is monotonically decreasing, $\forall x < c^*, \psi(x) > \psi(c^*) \geq 0$, so c^* is feasible.

Proof for Case (ii). Since $\psi'(c)$ is separable, there exists c' such that $\psi'(c) > 0$ on $(1, c')$ and $\psi'(c) < 0$ on (c', ∞) . By Newton-Leibnitz formula: $\psi(c') = \psi(1) + \int_1^{c'} \psi'(t) dt$, so $\psi(c') \geq \psi(1) \geq 0$. Since $\psi'(c) < 0$ on (c', ∞) , then by similarly to (i), we conclude that ψ is separable.

Proof for Case (iii). If $\psi/(c^a)$ is separable for some $a \in \mathbb{R}$, then there exists c^* such that $\psi(c)/c^a > 0$ on $(1, c^*)$ and $\psi(c)/c^a < 0$ on (c^*, ∞) , since $c^a > 0$ for $c > 1$, we conclude that $\psi(c) > 0$ on $(1, c^*)$ and $\psi(c) < 0$ on (c^*, ∞) , so ψ is separable.

C SUPPLEMENTARY MATERIAL FOR PROBABILISTIC RECURRENCES

C.1 On the concentration bound for Quicksort

Here we emphasize use exactly the same step in our paper to derive the following bound for all $k_1, k_2 > 0$:

$$\Pr[\mathcal{T}(n^*) \geq (9 + k_1) \cdot n^* \ln n^* + k_2 \cdot n] \leq (2.3)^{-k_1 \cdot \ln n^* - k_2}$$

The steps below until the sentence "Finally, we could prove.." are exactly the same as our paper in Page 14, without changing any word.

Given $f(n) := 9 \cdot n \cdot \ln n$, we apply Lemma 5.3 and obtain the following conditions for α :[‡]

$$\forall 1 \leq n \leq n^*, \alpha^{9 \cdot n \cdot \ln n} \geq \alpha^{n-1} \cdot \frac{1}{n} \cdot \sum_{i=0}^{n-1} \alpha^{f(i)+f(n-1-i)}$$

For $1 \leq n \leq 8$, we can manually verify that it suffices to set $\alpha > 1$. In the sequel, we assume $n \geq 8$. Note that $(i \cdot \ln i + (n - i - 1) \cdot \ln(n - i - 1))$ is monotonically decreasing on $[1, \lfloor n/2 \rfloor]$, and monotonically increasing on $[\lceil n/2 \rceil, n]$. We partition the summation above uniformly into eight parts and use the maximum of each part to overapproximate the sum. This leads to the following upper bound for this summation:

$$\begin{aligned} \sum_{i=0}^{n-1} \alpha^{f(i)} &\leq \sum_{j=0}^7 \sum_{i=\lceil j \cdot n/8 \rceil}^{\lfloor (j+1) \cdot n/8 \rfloor} \alpha^{f(i)+f(n-1-i)} \\ &\leq \frac{n}{4} \cdot \left(\alpha^{9 \cdot n \cdot \ln n} + \alpha^{9 \cdot (\frac{n}{8} \cdot \ln \frac{n}{8} + \frac{7 \cdot n}{8} \cdot \ln \frac{7 \cdot n}{8})} + \alpha^{9 \cdot (\frac{n}{4} \cdot \ln \frac{n}{4} + \frac{3 \cdot n}{4} \cdot \ln \frac{3 \cdot n}{4})} + \alpha^{9 \cdot (\frac{5 \cdot n}{8} \cdot \ln \frac{5 \cdot n}{8} + \frac{3 \cdot n}{8} \cdot \ln \frac{3 \cdot n}{8})} \right) \end{aligned}$$

Plugging in this overapproximation back into the original inequality, we get:

$$\alpha^{9 \cdot n \cdot \ln n} \geq \alpha^{n-1} \cdot \frac{1}{4} \cdot \left(\alpha^{9 \cdot n \cdot \ln n} + \alpha^{9 \cdot (\frac{n}{8} \cdot \ln \frac{n}{8} + \frac{7 \cdot n}{8} \cdot \ln \frac{7 \cdot n}{8})} + \alpha^{9 \cdot (\frac{n}{4} \cdot \ln \frac{n}{4} + \frac{3 \cdot n}{4} \cdot \ln \frac{3 \cdot n}{4})} + \alpha^{9 \cdot (\frac{5 \cdot n}{8} \cdot \ln \frac{5 \cdot n}{8} + \frac{3 \cdot n}{8} \cdot \ln \frac{3 \cdot n}{8})} \right)$$

for all $8 \leq n \leq n^*$. We define $c := \alpha^n$ to do substitution, and we use the following formula to do strengthening:

$$\begin{aligned} \alpha^{\beta \cdot \ln \beta + (n-\beta) \cdot \ln(n-\beta)} &\leq \alpha^{\beta \cdot \ln n + (n-\beta) \cdot \ln(n-\beta)} \\ &= \alpha^{n \cdot \ln n} \cdot \alpha^{-(n-\beta) \cdot \ln n + (n-\beta) \cdot \ln(n-\beta)} \\ &= \alpha^{n \cdot \ln n + (n-\beta) \cdot \ln \frac{n-\beta}{n}} = c \cdot c^{\frac{(n-\beta)}{n \cdot \ln n} \cdot \ln \frac{n-\beta}{n}} \end{aligned}$$

[‡]We assume $0 \ln 0 := 0$.

By defining $\beta = \frac{n}{8}, \frac{n}{4}, \frac{3n}{8}$ respectively, we obtain:

$$\begin{aligned} \forall 8 \leq n \leq n^*, c^{9 \ln n} &\geq \frac{c}{n} \cdot \frac{n}{4} \cdot \left(c^{9 \ln n} + c^{9 \ln n + \frac{63}{8} \cdot \ln \frac{7}{8}} + c^{9 \ln n + \frac{27}{4} \cdot \ln \frac{3}{4}} + c^{9 \ln n + \frac{45}{8} \cdot \ln \frac{5}{8}} \right) \\ \forall 8 \leq n \leq n^*, 4 - \left(c + c^{1 - \frac{63}{8} \cdot \ln \frac{8}{7}} + c^{1 - \frac{27}{4} \cdot \ln \frac{4}{3}} + c^{1 - \frac{45}{8} \cdot \ln \frac{8}{5}} \right) &\geq 0 \end{aligned}$$

Now we study the following function:

$$\psi(c) = 4 - \left(c + c^{1 - \frac{63}{8} \cdot \ln \frac{8}{7}} + c^{1 - \frac{27}{4} \cdot \ln \frac{4}{3}} + c^{1 - \frac{45}{8} \cdot \ln \frac{8}{5}} \right).$$

By basic calculus, we can prove that $\psi(c) \geq 0$ holds on $[1, 2.3]$. Additionally, since for every α , α^n increases as n increases, by plugging $\alpha = 2.3^{1/(n^*)}$ into Lemma 5.3, we obtain:

$$\Pr[\mathcal{T}(n^*) \geq (9 + k_1) \cdot n^* \ln n^* + k_2 \cdot n] \leq (2.3)^{-k_1 \cdot \ln n^* - k_2}$$