On the Complexity of Restarting

Jan-Hendrik Lorenz(✉)

Institute of Theoretical Computer Science, Ulm University, 89069 Ulm, Germany
jan-hendrik.lorenz@uni-ulm.de

Abstract. Restarting is a technique used by many randomized local search and systematic search algorithms. If the algorithm has not been successful for some time, the algorithm is reset and reinitialized with a new random seed. However, for some algorithms and some problem instances, restarts are not beneficial. Luby et al. [12] showed that if restarts are useful, then there is a restart time t^* such that the so-called fixed-cutoff strategy is the best possible strategy in expectation.

In this work, we show that deciding whether restarts are useful is NP-complete. Furthermore, we show that there is no feasible approximation algorithm for the optimal restart time t^*. Lastly, we show that calculating the expected runtime for a known probability distribution and a given restart time is #P-complete.

Keywords: Restarts · Fixed-cutoff strategy · Probability distribution · Computational complexity · NP · #P · Inapproximability

1 Introduction

Some (randomized) algorithms employ an algorithmic paradigm called restarting: If a solution is not found after a certain number of steps, then the algorithm is reset and the search starts over with a new random seed. Restarts are especially prevalent in stochastic local search (e.g. [13,17]) and randomized systematic search algorithms (e.g. [3,9]). In practice, restarts help to improve the performance of some algorithms by orders of magnitude, and are presently employed in most state-of-the-art SAT solvers [4].

For Las Vegas algorithms the runtime behavior is often modeled with probability distributions. For example, Arbelaez et al. [1] use lognormal and exponential distributions to describe the runtime behavior of two SAT solvers. Frost et al. [8] model the runtime behavior of solvable CSP instances with Weibull distributions and the behavior of unsolvable CSP instances with lognormal distributions. Both papers use empirical observations to fit these distributions to the observed runtimes. Hence, it also seems natural for theoretical purposes to model the runtime behavior with cumulative distribution functions (cdfs) or probability mass functions (pmfs). It allows to succinctly represent the runtime behavior on a large (possibly infinite) support.

R. van Bevern and G. Kucherov (Eds.): CSR 2019, LNCS 11532, pp. 250–261, 2019.
https://doi.org/10.1007/978-3-030-19955-5_22

From the theoretical perspective, Luby et al. [12] showed that the so-called fixed-cutoff strategy is an optimal strategy w.r.t. the expected runtime for some restart time t^*. The fixed-cutoff strategy allows an unbounded number of restarts and the algorithm always restarts after t^* steps. It is, however, necessary to have nearly complete knowledge of the runtime behavior to compute the optimal restart time t^*.

A good restart strategy should improve the performance of the algorithm in question. For the case when the distribution is known, Lorenz [11] derived formulas to evaluate whether there is any restart strategy which is beneficial w.r.t. the expected runtime. Furthermore, the formulas can be used to calculate an optimal restart time.

Another crucial property for restart strategies is that the calculation of restart times should be efficient. In other words, the time spent choosing a restart strategy should be significantly less than the time dedicated to the actual algorithm that uses the restart strategy. So far, the complexity of choosing a restart strategy has not been considered.

Our Contribution: We study the computational complexity of problems directly related to restarts. It is assumed that the probability distribution is known and provided in symbolic form, i.e., the formulas of either the cdf or pmf of the distribution are known. Section 3 considers decision problems. It is shown that deciding whether there is a restart time such that the expected runtime is less than a threshold is NP-hard (Theorem 2) while a relaxation of the problem is still NP-complete (Theorem 1). Furthermore, Theorem 3 and Corollary 2 show that there is no feasible approximation algorithm for the restart time w.r.t. the expected runtime.

In Sect. 4, we investigate underlying reasons why the decision problems mentioned above are hard. Moorsel and Wolter [14] showed that restarts are beneficial if and only if $E[X] < E[X - t \mid X > t]$ for some t, where X is a random variable describing the runtime. Here, we show that calculating both sides of the inequality is #P-complete (Theorems 4 and 5).

Related Work: Initiated by the work of Batu et al. [2], *property testing* gained traction in recent years. In this field, the general objective is determining whether a probability distribution fulfills a specific desired property. For example, an algorithm should decide whether the pmf of a distribution is unimodal [7]. For an in-depth overview of the topic, refer to [6]. The difference to our model is that in the framework of property testing the probability distribution is not explicitly known. Instead, it is assumed that sample access to the distribution is available. In other words, there is an oracle which returns an element x with probability $p(x)$. Therefore, the pmf p cannot be observed directly, while in our model p is known. To the best of our knowledge, complexity properties of probability distributions provided in symbolic form have not been studied before.

2 Preliminaries

We presume that the reader is familiar with the basic notions of complexity theory. For a detailed description of the topic, we refer to a standard book on the topic like [16]. We assume that the runtime distribution is known and provided as input in symbolic form. First, the structure of the functions is specified: In this work, it is only required that a function F is well-defined on a bounded interval $I = \{0, \ldots, a\}$ with $0 < a < \infty$. In this work, we consider discrete cumulative distribution functions and probability mass functions.

Definition 1. *Let X be an integer-valued random variable. Then, the **cumulative distribution function (cdf)** F_X of X is defined by*

$$F_X(i) = \Pr(X \leq i).$$

*The **probability mass function (pmf)** p_X of X is defined by*

$$p_X(i) = \Pr(X = i).$$

In the following, the subscripts are often omitted. Cdfs and pmfs are bounded from below by zero and bounded from above by one. Moreover, cdfs are monotone on I. Let p be a pmf then its corresponding cumulative function $F(k) = \sum_{i=0}^{k} p(i)$ is required to be a cdf.

Each function F is defined with binary encoded input, i.e., $F : \{0,1\}^n \mapsto [0,1]$ where n is a suitable integer. The function F is given in symbolic form or in other words: as a formula. We use uniform families of arithmetic circuits to describe the formulas. In an arithmetic circuit, the gates are arithmetic operations, like addition or multiplication. An overview of arithmetic circuits can be found in [19]. We use binary encoded integers as input. Also, there is a large number of non-standard functions which could be used to achieve a succinct representation of an arbitrary function. Therefore, we only allow the use of addition, subtraction, multiplication, exponentiation, and division gates. Another work using a model with the same gates is, for example, described in [5]. Another significant property which is required for the following results is that $F(x)$ can be evaluated in polynomial time for all x. In the following, when we refer to formulas, we implicitly mean formulas which are given as arithmetic circuits.

In this work, it is often necessary to encode a conjunctive normal form (CNF) formula as a function G using only multiplication and subtraction such that the SAT formula is satisfiable iff $G(a) = 1$ for some input a. It is known that negation $\neg x$ can be expressed by $1 - x$, a logical `and` $x \wedge y$ is given by xy, and a logical `or` $x \vee y$ is $1 - (1 - x)(1 - y)$.

Here, we formally define the fixed-cutoff strategy.

Definition 2 ([12]). *Let $\mathcal{A}(x)$ an algorithm $\mathcal{A}$ on input x. Let t be a positive integer. A modified algorithm $\mathcal{A}_t$ is obtained by introducing a counter $T = 0$. Then, $\mathcal{A}(x)$ is allowed to run. Increment T after each step of $\mathcal{A}(x)$. If T exceeds t reset and reinitialize $\mathcal{A}(x)$ with a new random seed and set T to zero. If at*

any point $\mathcal{A}(x)$ finds a solution, then $\mathcal{A}_t(x)$ returns this solution. Repeat these steps until $\mathcal{A}(x)$ finds a solution. The integer t is called ***restart time****. This algorithmic approach is called* ***fixed-cutoff strategy****.*

In this work, we only address fixed-cutoff strategies. Hence, whenever we refer to restart strategies or restarts, we implicitly mean a fixed-cutoff strategy.

Let X be a discrete random variable with cdf F and let $t \in \mathbb{N}$ be an arbitrary restart time. The expected value with restarts after t steps is denoted by $E[X_t]$. Luby et al. [12] showed that $E[X_t]$ is given by

$$E[X_t] = \frac{1 - F(t)}{F(t)} t + E[X \mid X \leq t] = \frac{1}{F(t)} \left(t - \sum_{x<t} F(x) \right). \tag{1}$$

They also introduced some useful bounds in their work:

Lemma 1 ([12]). *Let $l^* = \inf_t E[X_t]$ be the optimal expected value under restart, then*

$$l^* \leq \min_t \frac{t}{F(t)} \leq 4l^* \tag{2}$$

holds.

For the rest of this work, the quotient $t/F(t)$ is called the upper bound (of the expected value with restarts).

3 Hardness and Inapproximability

We first focus on the upper bound $t/F(t)$ and show that this version of the problem is computationally hard.

RESTART-UPPER-BOUND-CDF
Input: A cdf $F : \mathbb{N} \mapsto [0, 1]$ and an integer k. The cdf F is given as an arithmetic circuit and $F(t)$ can be evaluated in polynomial time for every $t < k$. The symbolic form of F must be composed of summation, subtraction, multiplication and division operations.
Question: Is there a restart time t such that $\frac{t}{F(t)} < k$?

Theorem 1. RESTART-UPPER-BOUND-CDF *is NP-complete.*

Proof. We first show that RESTART-UPPER-BOUND-CDF is in NP. It is easy to see that the inequality $\frac{t}{F(t)} < k$ only holds if $t < k$. Then, the inequality $\frac{t}{F(t)} < k$ can be verified in polynomial time since by hypothesis F can be evaluated in polynomial time for every fixed $t < k$. Thus, RESTART-UPPER-BOUND-CDF is in NP.

Secondly, let G be any SAT formula with n variables. Any $0 \leq t \leq 2^n - 1$ can be interpreted as an assignment of the variables in G; this assignment is denoted

by α_t. In the following, $G\alpha$ denotes the numeric value of G evaluated with the assignment α. We define the function $F_G : \{0,1\}^{n+1} \mapsto [0,1]$:

$$F_G(t) = \begin{cases} \frac{t+G\alpha_t}{2^n}, & 0 \leq t \leq 2^n - 1 \\ 1, & t \geq 2^n \end{cases} \tag{3}$$

The function $F_G(t)$ can be evaluated in polynomial time for all $t \leq 2^n$ because the division by 2^n can be expressed in polynomial length with the standard binary representation. Clearly, F_G is bounded from below by zero and from above by one. The formula F_G is monotonically increasing which can be derived from the fact that $G\alpha_t \leq 1 + G\alpha_{t+1}$ holds.

$$\begin{aligned} & G\alpha_t \leq 1 + G\alpha_{t+1} \\ \Leftrightarrow & \frac{t + G\alpha_t}{2^n} \leq \frac{t + 1 + G\alpha_{t+1}}{2^n} \end{aligned}$$

What remains to be shown is that F_G can be expressed with summation, subtraction, multiplication and division. Each of the two cases can clearly be expressed with these operations, since a CNF formula can be expressed as a function as described above. Only the case distinction has to expressed with the allowed operations. This can be achieved with the following representation:

$$F_G(t) = \frac{t + G\alpha_t}{2^n} \cdot (1 - x_{n+1}) + x_{n+1}.$$

Where x_{n+1} is the $(n+1)$-st bit of the binary representation of t. Hence, F_G is a cumulative distribution function which can be evaluated in polynomial time for each $t < 2^n + 1$. Let U be an unsatisfiable formula, then $F_U(t) = \frac{t}{2^n}$ for all t with $0 \leq t \leq 2^n$. Therefore,

$$\frac{t}{F_U(t)} = 2^n$$

holds for all $t \leq 2^n$. On the other hand, let S be any satisfiable formula and let x be any integer such that α_x is a satisfying assignment of S. Then,

$$\frac{x}{F_S(x)} = \frac{x}{x+1} 2^n < 2^n.$$

A SAT instance G is satisfiable if and only if there is a t with $t/F_G(t) < 2^n$. Therefore, Restart-upper-bound-cdf is NP-complete. □

The version shown above is for the computation of an upper bound of the expected value under restart. This raises the question whether the exact version is also computationally hard. The next section addresses this problem. For this, the used function is also allowed to use the exponentiation operation.

Restart-exact-cdf

Input: A cdf $F : \mathbb{N} \mapsto [0,1]$ and an integer k. The arithmetic circuit of F must be composed of summation, subtraction, multiplication, division and exponentiation operations.

Question: Is there a restart time t such that $E[X_t] < k$?

Theorem 2. *The problem* Restart-exact-cdf *is NP-hard.*

Proof. Let G and α_t be defined as in the proof of Theorem 1 and define the function F_G:

$$F_G(t) = \begin{cases} 0, & t < 1 \\ 1 - 2^{-t-G\alpha_{t-1}}, & 1 \leq t \leq 2^n \\ 1 - 2^{-t}, & t > 2^n \end{cases} \tag{4}$$

The value of $F(t)$ in standard binary representation is $0.11\ldots1$ with $t + G\alpha_{t-1}$ ones in the decimal places. This can be exponential in the length of the input. However, the value of $F(t)$ can be encoded unambiguously by just saving the number of ones in the decimal places. Thus, a representation of $F(t)$ can be computed in polynomial time. The function F is a cdf which can be verified in the same way as in Theorem 1. We define a random variable X such that F is the cdf of X. If U is an unsatisfiable SAT instance, then F_U is a geometric distribution with success probability 0.5. For this geometric distribution, it is known that $E[X_t] = 2$ for every $t \in \mathbb{N}$. If S is a satisfiable SAT instance and t is an integer such that $G\alpha_{t-1} = 1$, then the expected value $E[X_t]$ is at most

$$E[X_t] \leq 2 \cdot \frac{1 - 2^{-t}}{1 - 2^{-t-1}} < 2.$$

Hence, a SAT instance is satisfiable if and only if there is a t with $E[X_t] < 2$. □

Instead of the cdf, it is also possible to use the pmf for both problems. Let Restart-upper-bound-pmf and Restart-exact-pmf denote those problems. I.e., instead of a cdf the input consists of an arithmetic circuit calculating a pmf.

Corollary 1. Restart-upper-bound-pmf *and* Restart-exact-pmf *are NP-hard.*

Proof. Let F be a cdf which can be evaluated in polynomial time and which is given as an arithmetic circuit. Then, $p(t) = F(t) - F(t-1)$ is a pmf which can be expressed as an arithmetic circuit. This can be used as a reduction from Restart-upper-bound-cdf to Restart-upper-bound-pmf and respectively from Restart-exact-cdf to Restart-exact-pmf. □

The results shown so far are for the decision whether a restart strategy should be applied. Another important problem concerning restarts is the choice of a good restart time. In the following, we show that restart times cannot be approximated in polynomial time up to an arbitrary constant c unless P = NP. To make the notion more precise: Let X be a discrete random variable and let t^* be an optimal restart time such that the optimal expected value under restart is $E[X_{t^*}]$. Then, there is no polynomial-time algorithm which finds a restart time t such that $E[X_t] \leq c \cdot E[X_{t^*}]$. First, we show that the upper bound $t/F(t)$ is also inapproximable.

Theorem 3. *Let X denote a discrete random variable, p the pmf of X and $\frac{t^*}{F(t^*)}$ its optimal upper bound for the expected value under restart. Let $c > 1$ be any arbitrary constant. Unless $P = NP$, there is no polynomial-time algorithm $\mathcal{A}(p)$ such that $\mathcal{A}(p)$ finds a restart time t such that the upper bound $\frac{t}{F(t)}$ is at most $c\frac{t^*}{F(t^*)}$ with p as input.*

Proof. Let G and α_t be defined as in the proof of Theorem 1. We define the function p_G:

$$p_G(t) = \begin{cases} \frac{G\alpha_{t-1}}{2^n+1}, & 1 \leq t \leq 2^n \\ \frac{1}{2^n+1}, & t = c \cdot 2^n \\ 0, & \text{else} \end{cases} \tag{5}$$

The function p_G is a pmf since it is non-negative for every t and sums up to at most one. Let U be an unsatisfiable SAT instance and let $F_U(t) = \sum_{i=0}^{t} p_U(i)$ be the cdf which corresponds to p_U. Let l_U denote the minimum of the upper bound $t/F_U(t)$. The value of $F_U(t)$ is greater than zero iff $t \geq c2^n$. Thus, the upper bound is minimal at $t = c2^n$:

$$l_U = \min_t \frac{t}{F_U(t)} = c(2^n + 1)2^n.$$

On the other hand, let S be a satisfiable SAT instance and, again, let F_S be the cdf which corresponds to p_S. Let l_S denote the minimum of the upper bound $t/F_S(t)$. The upper bound is minimal for t such that there is no $x < t$ with $S\alpha_{x-1} = 1$. Then, $(2^n + 1)2^n$ bounds l_S from above:

$$l_S = \min_t \frac{t}{F(t)} \leq \frac{2^n}{F(2^n)} \leq (2^n + 1)2^n.$$

Hence, a SAT instance G is satisfiable if and only if there is a restart time t with $t/F_G(t) < c(2^n + 1)2^n$.

Let $\mathcal{A}$ be a polynomial-time algorithm which takes a pmf as input such that $\mathcal{A}(p)$ finds a restart time x with

$$\frac{x}{F(x)} \leq (c - \varepsilon) \min_t \frac{t}{F(t)},$$

where ε is an arbitrary constant with $0 < \varepsilon < c$. Consider the properties of $\mathcal{A}(p_S)$: The algorithm finds a restart time x such that

$$\frac{x}{F(x)} \leq (c - \varepsilon)l_S \leq (c - \varepsilon)(2^n + 1)2^n < c(2^n + 1)2^n = l_U.$$

As the constant c can be chosen arbitrarily, the existence of any polynomial-time approximation algorithm implies that checking whether the predicted restart time is less than $c \cdot 2^n$ decides SAT in polynomial time. □

Corollary 2. *Let X denote a discrete random variable, p the pmf of X and $E[X_{t^*}]$ its optimal expected value under restart. Let $c > 1$ be any arbitrary constant. Unless $P = NP$, there is no polynomial-time algorithm $\mathcal{A}(p)$ which always finds a restart time t such that the expected value under restart $E[X_t]$ is at most $cE[X_{t^*}]$ with p as input.*

The proof is analogous to the proof of Theorem 3 and is therefore omitted.

4 The Hardness of the Mean

Valiant [18] introduced the complexity class #P. The following, equivalent definition is from [20].

Definition 3 ([18], [20]). *Let M be any non-deterministic Turing machine and let $\mathtt{acc}_M(x)$ denote the number of accepting computations of M on input x. Then $\#P$ is given by*

$$\#P = \{f \mid f : \Sigma^* \mapsto \mathbb{N} \text{ such that } f = \mathtt{acc}_M \text{ for some polynomial time non-deterministic Turing machine } M\}.$$

In the following, we use *metric reductions* as defined by Krentel.

Definition 4 ([10]). *Let f, h be functions. The function f is **metrically reducible** to h if and only if there are two functions g_1, g_2 which are computable in polynomial time such that $f(x) = g_2(x, h(g_1(x)))$.*

The formal definition of #P-hardness and completeness requires Cook reductions. However, polynomial-time many-one reductions and metric reductions both imply the existence of Cook reductions (compare [15]). Thus, the formal definition of Cook reductions and #P-hardness is omitted. We refer the reader to [18] for both definitions. A canonical $\#P$-complete problem is finding the number of satisfying assignments for a given boolean formula [18]. For the rest of this work, this problem is called #SAT.

#SAT
Input: A boolean formula F in conjunctive normal form.
Question: What is the number of satisfying assignments for F?

In the following, $\#\mathrm{SAT}(G)$ denotes the number of satisfying assignments of a SAT formula G in CNF. Let $\mathcal{A}$ be an algorithm on some input x and let X be a random variable describing the runtime distribution of $\mathcal{A}(x)$. Moorsel and Wolter [14] showed that $\mathcal{A}(x)$ benefits from restarts if and only if there is a $t > 0$ with $E[X] < E[X - t \mid X > t]$. Moreover, the expected runtime $E[X_t]$ with restarts requires the conditional mean $E[X \mid X \leq t]$ (compare Eq. 1). Hence, computing the expected value $E[X]$ and the conditional expected values $E[X \mid X > t]$ and $E[X \mid X \leq t]$ is an important task in the context of restarts.A relevant step in computing the conditional expected value is calculating partial moments: $\mu_X(N) = \sum_{i=0}^{N} i \Pr(X = i)$. This is because $E[X \mid X \leq N] = \frac{\mu_X(N)}{\Pr(X \leq N)}$ and

$E[X \mid X > N] = \frac{E[X]-\mu_X(N)}{\Pr(X>N)}$. More generally, the k-th partial moment $\mu_X^{(k)}(N)$ is defined as:

$$\mu_X^{(k)}(N) = \sum_{i=0}^{N} i^k \Pr(X = i).$$

Here, we address the computational complexity of calculating partial moments. First, we study the complexity of computing the partial expectation $\mu_X^{(1)}(N)$. To achieve a better fit to standard computational models we restrict the allowed functions such that each cdf can easily be mapped to integers. More precisely, only functions of the form $F(i) = \frac{c(i)}{M}$ for some natural number M and some function $c : \mathbb{N} \mapsto \mathbb{N}$ are considered. Then, $ME[X]$ is computed instead of the expected value itself.

Partial-expectation-cdf

Input: An integer N, a polynomial-time function $c : \{0, \ldots, N\} \mapsto \mathbb{N}$ and an integer M, such that $F(i) = \frac{c(i)}{M}$ is a (partial) cdf.

Question: What is the partial expectation $M\mu_X^{(1)}(N)$, where F defines X.

Here, a partial cdf denotes a function which is monotonically increasing and does not exceed 1.

Theorem 4. Partial-expectation-cdf *is #P-complete with respect to metric reductions.*

Proof. First, we show that Partial-expectation-cdf is in #P. Note that F has bounded support due to its definition. The partial expectation is then given by

$$M\mu_X^{(1)}(N) = M\sum_{i=1}^{N}(1 - F(i)) = \sum_{i=1}^{N}(M - c(i)).$$

We design a non-deterministic polynomial time Turing machine T such that the number of accepting computation paths of T is equal to $M\mu_X^{(1)}(N)$. The Turing machine T takes c, N, M as input. A positive integer $i \in \{0, \ldots, N\}$ is chosen non-deterministically and $c(i)$ is evaluated. Then, add $M - c(i)$ accepting paths to T which are identified by additional non-deterministic bits. For this, observe that the difference $M - c(i)$ is a positive number since F is a (partial) cdf. The total number of accepting paths of T is then $\sum_{i=1}^{N}(M - c(i)) = M\mu_X^{(1)}(N)$ which shows that Partial-expectation-cdf is in #P.

Next, each $c(i)$ is bounded by M because otherwise, F would not be a cdf. Therefore, the computation of $M\mu_X^{(1)}(N)$ is in #P. In the following, we reduce #SAT to Partial-expectation-cdf to show the #P-hardness. Let G and α_t be defined as in the proof of Theorem 1. The function $F_G(t)$ is given by:

$$F_G(t) = \begin{cases} \frac{(t-1)+G\alpha_{t-1}}{2^n}, & 1 \leq t \leq 2^n \\ 1, & t > 2^n \end{cases} \tag{6}$$

Let X be the random variable defined by F_G. Here, M is equal to 2^n and N is (at most) 2^n, the function c is given by $c(t) = (t-1) + G\alpha_{t-1}$ for $t \leq 2^n$ and $c(t) = 2^n$ for $t > 2^n$. Note that $\mu_X^{(1)}(N)$ is equal to the expected value $E[X]$. Observe the value of $2^n E[X]$:

$$\begin{aligned} 2^n E[X] &= \sum_{i=1}^{2^n} 2^n - c(i) = 2^{2n} - \sum_{i=1}^{2^n} (i-1) + G\alpha_{i-1} \\ &= 2^{2n} - 2^{n-1}(2^n - 1) - \sum_{i=0}^{2^n-1} G\alpha_i \\ &= 2^{2n} - 2^{n-1}(2^n - 1) - \#SAT(G). \end{aligned}$$

Therefore, #SAT can be metrically reduced to Partial-expectation-cdf by encoding the SAT instance as a cdf as in F_G for g_1 and setting $g_2(G,k)$ to $-\left(k - 2^{2n} + 2^{n-1}(2^n - 1)\right)$. □

The problem can also be defined with probability mass functions as input.

Partial-moment-pmf

Input: An integer N, a polynomial-time function $c : \{0, \ldots, N\} \mapsto \mathbb{N}$, an integer k and an integer M, such that $p(i) = \frac{c(i)}{M}$ is a (partial) pmf and such that $i^k \cdot c(i)$ is a natural number for all i.

Question: What is the partial moment $M\mu_X^{(k)}(N)$, where F defines X.

Theorem 5. Partial-moment-pmf *is #P-complete with respect to polynomial-time many-one reductions.*

Proof. We show that Partial-moment-pmf is in #P. The partial moment $M\mu_X^{(k)}(N)$ is given by $M\sum_{i=0}^{N} i^k p(i)$. By definition, $Mi^k p(i)$ is a natural number which can be computed in polynomial time. Therefore, Partial-moment-pmf is in #P.

Let G and α_t be defined as in the proof of Theorem 1.

$$p_G(t) = \begin{cases} \frac{G\alpha_{t-1}}{2^n \cdot t^k}, & 1 \leq t \leq 2^n \\ 0, & \text{else} \end{cases} \tag{7}$$

Here, M and N are 2^n and the function $c(t)$ is given by $\frac{G\alpha_{t-1}}{t^k}$. The function $p_G(t)$ is a partial pmf since the sum $\sum_{t=1}^{2^n} \frac{G\alpha_{t-1}}{2^n \cdot t^k}$ does not exceed 1 for all positive k. Clearly, the product $t^k c(t) = G\alpha_{t-1}$ is a natural number for all t with $1 \leq t \leq 2^n$. The partial expectation $2^n \mu_X^{(k)}(2^n)$ is given by

$$2^n \mu_X^{(k)}(2^n) = \sum_{i=1}^{2^n} i^k \cdot c(i) = \sum_{i=1}^{2^n} G\alpha_{i-1} = \#\mathrm{SAT}(G).$$

Therefore, #SAT is polynomial-time many-one reducible to Partial-moment-pmf which completes the proof. □

We conclude that the computations required to decide $E[X] < E[X - t \mid X > t]$ and to calculate $E[X_t]$ are #P-complete.

5 Conclusion and Outlook

There are three major questions related to restarts in randomized algorithms:

1. Should the algorithm use a restart strategy?
2. If yes, when should it restart?
3. What is the expected runtime of the restarted process?

In this work, we evaluate the computational complexity of all three problems. We assume that the formula of the underlying probability distribution is known. I.e., the formula of either the probability mass function (pmf) or the cumulative distribution function (cdf) is used as input.

Deciding whether a restart strategy should be used is NP-hard (Theorems 1, 2 and Corollary 1). Finding a good restart time is addressed for the case when the pmf is known.

Theorem 3 and Corollary 2 show that there is no polynomial-time algorithm which only uses the pmf and has the following properties: The algorithm computes a restart time such that its corresponding expected runtime is worse by only a constant factor compared to the best restart strategy. Lastly, an essential step for computing the expected runtime with restarts is the computation of the so-called partial expectation. We show that the computation of the partial expectation is #P-complete (Theorems 4 and 5).

There are some loose ends in this work: We showed that there is no feasible approximation algorithm for the restart time if the pmf is known. However, the question whether there is an approximation scheme in the case when the cdf is known remains unanswered. Furthermore, while the general problems presented in this work are hard, there might be subclasses of problems which can be solved in polynomial time. On the other hand, we showed that Restart-exact-cdf is NP-hard. It would be interesting to find out more about its computational complexity. We believe the problem could be undecidable if the support of the cdf is unbounded.

Moreover, we assume that the formulas provided as input describe cdfs or pmfs. Therefore, the properties proven in this work can be viewed in the context of promise problems. In fact, it is a non-trivial task to check whether the input formula is indeed either a cdf or a pmf.

Furthermore, to the best of our knowledge, the properties of probability distributions have not been studied before in the setting of computational complexity if the distribution is known as a formula. There are other attributes which could be analyzed, e.g., the hazard rate or the computation of quantiles.

References

1. Arbelaez, A., Truchet, C., O'Sullivan, B.: Learning sequential and parallel runtime distributions for randomized algorithms. In: ICTAI 2016, San Jose, California, USA, pp. 655–662. IEEE (2016)

2. Batu, T., Fortnow, L., Rubinfeld, R., Smith, W., White, P.: Testing that distributions are close. In: Proceedings of the Foundations of Computer Science, pp. 259–269. IEEE (2000)
3. Biere, A., Fröhlich, A.: Evaluating CDCL restart schemes. In: Proceedings of the International Workshop on Pragmatics of SAT (POS 2015) (2015)
4. Biere, A., Heule, M., van Maaren, H.: Handbook of Satisfiability. IOS Press, Amsterdam (2009)
5. Bshouty, D., Bshouty, N.H.: On interpolating arithmetic read-once formulas with exponentiation. J. Comput. Syst. Sci. **56**(1), 112–124 (1998)
6. Canonne, C.: A survey on distribution testing: your data is big. But is it blue? In: Electronic Colloquium on Computational Complexity (ECCC), vol. 22, no. 63 (2015)
7. Canonne, C., Diakonikolas, I., Gouleakis, T., Rubinfeld, R.: Testing shape restrictions of discrete distributions. Theor. Comput. Syst. **62**(1), 4–62 (2018)
8. Frost, D., Rish, I., Vila, L.: Summarizing CSP hardness with continuous probability distributions. In: Proceedings of the AAAI 1997/IAAI 1997, pp. 327–333. AAAI Press (1997)
9. Gomes, C., Selman, B., Kautz, H.: Boosting combinatorial search through randomization, pp. 431–437. AAAI Press (1998)
10. Krentel, M.: The complexity of optimization problems. J. Comput. Syst. Sci. **36**(3), 490–509 (1988)
11. Lorenz, J.-H.: Runtime distributions and criteria for restarts. In: Tjoa, A.M., Bellatreche, L., Biffl, S., van Leeuwen, J., Wiedermann, J. (eds.) SOFSEM 2018. LNCS, vol. 10706, pp. 493–507. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-73117-9_35
12. Luby, M., Sinclair, A., Zuckerman, D.: Optimal speedup of Las Vegas algorithms. Inf. Process. Lett. **47**(4), 173–180 (1993)
13. Mengshoel, O., Wilkins, D., Roth, D.: Initialization and restart in stochastic local search: computing a most probable explanation in bayesian networks. IEEE Trans. Knowl. Data Eng. **23**(2), 235–247 (2011)
14. Moorsel, A., Wolter, K.: Analysis and algorithms for restart. In: Proceedings of the First International Conference on the Quantitative Evaluation of Systems, pp. 195–204 (2004)
15. Pagourtzis, A., Zachos, S.: The complexity of counting functions with easy decision version. In: Královič, R., Urzyczyn, P. (eds.) MFCS 2006. LNCS, vol. 4162, pp. 741–752. Springer, Heidelberg (2006). https://doi.org/10.1007/11821069_64
16. Papadimitriou, C.: Computational Complexity. Addison Wesley Pub. Co., Boston (1994)
17. Schöning, U.: A probabilistic algorithm for k-SAT and constraint satisfaction problems. In: 40th Annual Symposium on Foundations of Computer Science, pp. 410–414 (1999)
18. Valiant, L.: The complexity of computing the permanent. Theoret. Comput. Sci. **8**(2), 189–201 (1979)
19. Vollmer, H.: Introduction to Circuit Complexity: A Uniform Approach. Texts in Theoretical Computer Science, An EATCS Series. Springer, Heidelberg (1999). https://doi.org/10.1007/978-3-662-03927-4
20. Welsh, D.: Complexity: Knots, Colourings and Countings. Cambridge University Press, New York (1993)