12-2020

Algorithms and hardness results for computing cores of Markov chains

Ali AHMADI

Krishnendu CHATTERJEE

Amir KAFSHDAR GOHARSHADY

Tobias MEGGENDORFER

Roodabeh SAFAVI

See next page for additional authors

Citation

AHMADI, Ali; CHATTERJEE, Krishnendu; KAFSHDAR GOHARSHADY, Amir; MEGGENDORFER, Tobias; SAFAVI, Roodabeh; and ZIKELIC, Dorde. Algorithms and hardness results for computing cores of Markov chains. (2020). *Proceedings of the 42nd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2022), Chennai, India, December 18-20*. 250, 1-20.

Available at: https://ink.library.smu.edu.sg/sis_research/9059

Author

Ali AHMADI, Krishnendu CHATTERJEE, Amir KAFSHDAR GOHARSHADY, Tobias MEGGENDORFER, Roodabeh SAFAVI, and Dorde ZIKELIC

Algorithms and Hardness Results for Computing Cores of Markov Chains

Ali Ahmadi
Hong Kong University of Science and Technology (HKUST), China

Krishnendu Chatterjee
Institute of Science and Technology Austria (ISTA), Klosterneuburg, Austria

Amir Kafshdar Goharshady
Hong Kong University of Science and Technology (HKUST), China

Tobias Meggendorfer
Institute of Science and Technology Austria (ISTA), Klosterneuburg, Austria

Roodabeh Safavi
Institute of Science and Technology Austria (ISTA), Klosterneuburg, Austria
Hong Kong University of Science and Technology (HKUST), China

Đorđe Žikelić
Institute of Science and Technology Austria (ISTA), Klosterneuburg, Austria

Abstract

Given a Markov chain $M = (V, v_0, \delta)$, with state space V and a starting state v_0, and a probability threshold ϵ, an ϵ-core is a subset C of states that is left with probability at most ϵ. More formally, $C \subseteq V$ is an ϵ-core, iff $\mathbb{P}\left[\mathsf{reach}\left(V \backslash C\right)\right] \leq \epsilon$. Cores have been applied in a wide variety of verification problems over Markov chains, Markov decision processes, and probabilistic programs, as a means of discarding uninteresting and low-probability parts of a probabilistic system and instead being able to focus on the states that are likely to be encountered in a real-world run. In this work, we focus on the problem of computing a minimal ϵ-core in a Markov chain. Our contributions include both negative and positive results: (i) We show that the decision problem on the existence of an ϵ-core of a given size is NP-complete. This solves an open problem posed in [26]. We additionally show that the problem remains NP-complete even when limited to acyclic Markov chains with bounded maximal vertex degree; (ii) We provide a polynomial time algorithm for computing a minimal ϵ-core on Markov chains over control-flow graphs of structured programs. A straightforward combination of our algorithm with standard branch prediction techniques allows one to apply the idea of cores to find a subset of program lines that are left with low probability and then focus any desired static analysis on this core subset.

2012 ACM Subject Classification Software and its engineering → Formal software verification; Theory of computation → Graph algorithms analysis

Keywords and phrases Markov Chains, Cores, Complexity

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2022.29

Funding The research was partially supported by the Hong Kong Research Grants Council ECS Project No. 26208122, ERC CoG 863818 (FoRM-SMArt), the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie Grant Agreement No. 665385, HKUST–Kaisa Joint Research Institute Project Grant HKJRI3A-055 and HKUST Startup Grant R9272. Ali Ahmadi and Roodabeh Safavi were interns at HKUST. Author names appear in alphabetical order.

42nd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2022).
Editors: Anuj Dawar and Venkatesan Guruswami; Article No. 29; pp. 29:1–29:20
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Markov Chains. Discrete-time Markov chains (MCs) are arguably the most classical and standard mathematical formalism for modelling randomness in discrete-time probabilistic systems [19]. In a nutshell, Markov chains comprise a set of states and a transition function, assigning to each state a distribution over successors. The system evolves by repeatedly drawing a successor state from the transition distribution of the current state. This can, for example, model communication over a lossy channel, a queuing network, or populations of predator and prey which grow and interact randomly. Indeed, Markov chains are an important tool in many areas, such as computer science [4], biology [30], epidemiology [21], and chemistry [20], to name a few. As such, analyzing MCs is an important question, and a central component of many modern model checkers, such as Prism [27] and Storm [17].

Markov Chains over Control-flow Graphs. An interesting family of MCs in verification are those that are defined over the control-flow graph (CFG) of a program [1], i.e. MCs in which the underlying graph, ignoring the probabilities, is a CFG. Such MCs can serve as formal semantics for simple probabilistic programs. They also appear naturally as a result of applying (statistical) branch prediction to a non-probabilistic program [33].

Markov Decision Processes. Extending MCs with non-determinism leads to the standard notion of Markov decision processes (MDPs) [31], which is the most common formalism for systems that exhibit both probabilistic and non-deterministic behavior. The non-determinism can model a wide variety of real-world phenomena such as the behavior of a controller, an unknown set of inputs, or simply abstractions made to ease the verification of the underlying system. Analysis of MDPs is a central topic in formal verification and model checking, see e.g. [2, 3, 4, 6, 10, 26].

Probabilistic Programs. Extending classical programs with the ability of generating random numbers by sampling from pre-defined distributions leads to the framework of probabilistic programs. Such programs have a wide variety of use-cases including in randomized algorithms [18, 29], stochastic network protocols [34], blockchain protocols and smart contracts [11, 13, 14], and robotics [23, 25]. Hence, they have been widely studied by the programming languages and verification communities [5, 24, 28, 7, 8, 22, 35, 15]. Probabilistic programs often have variables that can take integer or real values and their formal semantics are usually defined by an infinite-state MC, in absence of non-determinism, or an infinite-state MDP, in its presence [7, 36, 9].

Cores. Fixing a probability threshold $\epsilon \in (0, 1)$, an ϵ-core of a probabilistic system is a set of states that is left by a random run of the system with probability at most ϵ. Intuitively, when analyzing a large probabilistic system, one is interested in finding a core that is left with very low probability and then ignore the set of states that are outside this core, since it is unlikely that a real-world run of the system visits them. If the core happens to be much smaller than the original system, this simple idea can lead to huge improvements in the runtime of formal analyses. For example, suppose that we have applied branch prediction to a huge program and hence have probabilities associated to every transition in its CFG. By finding a small core, we can focus any static analysis on a set of lines of the program that actually matter, i.e. a set of lines that is left with low probability, and ignore the lines that are rarely or never encountered.

The concept of cores was introduced in [26] for the quantitative analysis of MCs and MDPs. Considering classical objectives such as mean-payoff, discounted sums of rewards, and hitting probabilities, [26] provided a partial-exploration framework that efficiently finds cores in large MDPs and then uses them to obtain not only a faster analysis, but also rigorous error bounds. An equivalent notion, called stochastic invariants, was introduced in [16] in the context of probabilistic programs and used in [12] to obtain quantitative bounds on their probability of termination.

Hardness of Computing a Core. Based on the discussion above, it is natural to aim for algorithms that find the smallest possible core in a probabilistic system, be it an MC, an MDP or a probabilistic program. In the case of probabilistic programs, any non-trivial formalization of this problem is undecidable as a direct result of the well-known Rice's theorem [32]. In the case of finite-state MDPs, given a threshold $\epsilon \in (0, 1)$ and an integer k, deciding whether the MDP has an ϵ-core of size k is an NP-complete problem [26]. However, the same problem was open when it comes to MCs [26], with neither efficient PTIME algorithms nor NP-hardness results provided so far.

Our Contribution. In this work, we consider the problem of finding an optimal ϵ-core in a finite-state discrete-time Markov chain and its decision variant, i.e. deciding whether an ϵ-core of size k exists. We obtain both hardness results and efficient algorithms:

- In Section 3, we prove that the decision problem is NP-complete. This settles the complexity for MCs and answers the open problem posed in [26]. We also show that the problem remains NP-hard even when limited to acyclic MCs with bounded degree.
- We then focus on positive results and provide a PTIME algorithm in Section 4 that is applicable to MCs over control-flow graphs.

In summary, our results show that computing cores in MCs is NP-hard in general and even over the very limited family of acyclic MCs with bounded degrees. Hence, in practice, cores should be computed using partial-exploration algorithms, as in [26], or other heuristics with no theoretical guarantees. However, in certain important use-cases, such as MCs over CFGs, we can compute an optimal core in PTIME. Notably, this is applicable to the problem of finding the core lines of a program given branch prediction data.

2 Preliminaries

We start by recalling the definitions of Markov chains and cores and fixing the notation that is used throughout this work.

Discrete Probability Distributions. Given a finite set X, a *probability distribution* over X is a function $\delta : X \to [0, 1]$ which satisfies the condition $\sum_{x \in X} \delta(x) = 1$. We use $\mathcal{D}(X)$ to denote the set of all probability distributions over X.

Markov Chains (MCs). A finite state *Markov chain* $M = (V, v_0, \delta)$ is an ordered triple consisting of a finite set of *states/vertices* V, a designated *initial state* $v_0 \in V$ and a *probabilistic transition function* $\delta : V \to \mathcal{D}(V)$, which, to each state in V assigns a probability distribution over its successor states. We define the set of *edges* of M as

$$E = \{(v, v') \in V \times V \mid \delta(v)(v') > 0\}.$$

Paths in Markov Chains. An infinite path in a Markov chain $M = (V, v_0, \delta)$ is an infinite sequence of states $\rho = (v_0, v_1, v_2, \dots)$ such that $\delta(v_i)(v_{i+1}) > 0$ holds for each $i \in \mathbb{N}$. Note that we require each infinite path to start in the initial state v_0 of the chain.

A Markov chain M admits a *probability measure* $\mathbb{P}^M$ over the set of all infinite paths in the Markov chain [31]. For each measurable set $O \subseteq V^\omega$ of infinite paths in M, we use $\mathbb{P}^M[O]$ to denote the probability of a random infinite path in the Markov chain being an element of the set O. In particular, if $T \subseteq V$ is a set of states, we use $\mathsf{reach}\, T$ to denote the set of all infinite paths in the Markov chain that visit at least one state in T, and use $\mathbb{P}^M[\mathsf{reach}\, T]$ to denote the probability of a random infinite path in the Markov chain reaching a vertex in T. When the Markov chain is clear from the context, we abbreviate the notation to $\mathbb{P}$, $\mathbb{P}[O]$ and $\mathbb{P}[\mathsf{reach}\, T]$.

Cores in Markov Chains. We now define the concept of cores in Markov chains which are the central object of study in this work. The notion of a core was originally defined in [26]. Given a probability threshold $\epsilon \in [0, 1]$, an *ϵ-core* in a Markov chain $M = (V, v_0, \delta)$ is a set of states $C \subseteq V$ such that a random infinite path in the Markov chain exits C with probability at most ϵ. More formally,

$$\mathbb{P}\Big[\mathsf{reach}\,(V \backslash C)\Big] \leq \epsilon.$$

For a natural number $k \in \mathbb{N}$, we say that $C \subseteq V$ is an (ϵ, k)-core if it is an ϵ-core of size at most k, i.e. $|C| \leq k$.

While the main negative result of this work is a proof of NP-completeness of the decision problem on whether a Markov chain contains an ϵ-core of at most a given size for $\epsilon \in (0, 1)$, we also strengthen this result by showing that the problem remains NP-complete even on a restricted class of Markov chains. We now formally define these restrictions.

Acyclic Markov Chains with Bounded Degree. A Markov chain $M = (V, v_0, \delta)$ naturally induces a directed graph $G_M = (V, E)$, where the vertex and the edge sets of the graph are defined by the set of states and the set of edges of the Markov chain. A Markov chain M is said to be *acyclic* if its induced graph G_M is acyclic, with the exception of a self-loop at one vertex. Note that the graph G_V cannot be completely acyclic, since it is a finite, directed graph and every state in a Markov chain must have at least one successor state for the probabilities of its outgoing edges to sum up to 1. Thus, a Markov chain must contain a proper cycle or a probability 1 self-loop. Therefore, acyclic Markov chains can be viewed as Markov chains whose induced graphs are "closest" to being acyclic, with the exception of a single probability 1 self-loop.

For a vertex/state $v \in V$, we define its *outdegree* to be the number of edges whose source vertex is v, i.e. $\mathsf{out}(v) = |\{u \in V \mid (v, u) \in E\}|$. Similarly, the *indegree* of v is defined as the number of edges whose target vertex is v, i.e. $\mathsf{in}(v) = |\{u \in V \mid (u, v) \in E\}|$. Together, we define the *degree* of a vertex v to be the total number of edges that are incident to v, i.e. $\mathsf{deg}(v) = \mathsf{out}(v) + \mathsf{in}(v)$. Finally, the *maximal vertex degree* of a Markov chain M is defined to be maximal degree of all its vertices, i.e. $\max_{v \in V} \mathsf{deg}(v)$.

In this work, we consider Markov chains which are both acyclic and have their maximal degree bounded by a constant.

3 Hardness Results

In this section, we show that the problem of computing the size of a minimal ϵ-core in a Markov chain is NP-hard for any non-trivial value of the probability threshold ϵ, i.e. any ϵ that is not equal to 0 or 1. Formally, given $\epsilon \in (0, 1)$, we prove in Theorem 1 that the problem of deciding whether a Markov chain M contains an (ϵ, k)-core for a given Markov chain M and a core size k is NP-complete. This implies that the problem of deciding whether a Markov chain contains an ϵ-core of size exactly k is also NP-complete, since one may always enlarge an (ϵ, k)-core in order to obtain an ϵ-core that contains exactly k states and so the two problems are immediately reducible to each other.

In [26], the authors proved NP-completeness of the problem for MDPs. However, it was hitherto not known whether computing cores of given size could be made more efficient in the case of Markov chains. This was left as an open problem in [26, Remark 3.7]. Our Theorem 1 solves the open problem of [26] and answers the posed question negatively. We conclude this section with Theorem 8, which shows that the problem of deciding the existence of a core of at most given size remains NP-complete even if we restrict it to acyclic Markov chains whose maximal vertex degree is at most 3.

▶ **Theorem 1** (Hardness of Finding a Core in an MC). *For an $\epsilon \in (0, 1)$, let $\textsc{Core}_\epsilon$ be the language defined as*

$$\Big\{(M, k) \mid M \textit{ is a Markov chain that contains an } (\epsilon, k)\textit{-core}\Big\}.$$

Then, $\textsc{Core}_\epsilon$ is NP-complete for any $\epsilon \in (0, 1)$.

Proof. To prove that $\textsc{Core}_\epsilon$ is contained in NP, note that the probability of reaching a given set of states in a Markov chain can be computed in polynomial time by reduction to a system of linear equations [4]. Hence, an (ϵ, k)-core can serve as its own witness of linear size. Thus, $\textsc{Core}_\epsilon$ is in NP.

To prove that $\textsc{Core}_\epsilon$ is NP-hard, we show a reduction from the $\textsc{Vertex-Cover}$problem which is a classical example of an NP-complete problem. Recall that, given an undirected graph $G = (V, E)$ and $k \in \mathbb{N}$, the $\textsc{Vertex-Cover}$problem is concerned with deciding if there exists a set of vertices $K \subseteq V$ of size k such that each edge in E is incident to at least one vertex in K.

Fix $\epsilon \in (0, 1)$ and consider an instance $(G = (V, E), k)$ of the $\textsc{Vertex-Cover}$problem where $n = |V| > \frac{1}{\epsilon}$ and $n \geq 4$. In order to obtain our reduction, we construct an instance $(M_{G,k}, k_{G,k})$ of the $\textsc{Core}_\epsilon$ problem where the Markov chain $M_{G,k}$ and natural number $k_{G,k}$ are both polynomial in the size of G and k and such that $(G, k) \in \textsc{Vertex-Cover}$if and only if $(M_{G,k}, k_{G,k}) \in \textsc{Core}_\epsilon$.

In the sequel, we say that a sequence of vertices $u_1, \ldots, u_s$ in a Markov chain (M, v, δ) form a *chain* of length s, if $\delta(u_i)(u_{i+1}) = 1$ for each $1 \leq i \leq s - 1$.

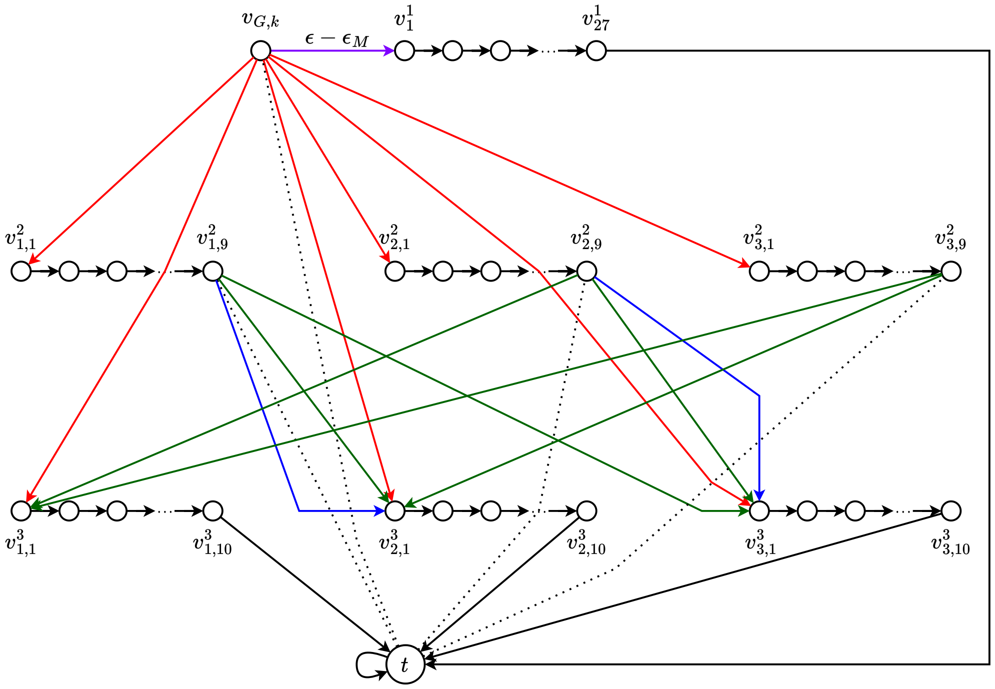

Figure 1 The figure depicts an example of a Markov chain $V_{G,k}$ constructed in the proof of Theorem 1 for a graph $G = (V, E)$ with $V = \{v_1, v_2, v_3\}$ and $E = \{(v_1, v_2), (v_2, v_3)\}$, and for an arbitrary value of k. Let $n = |V| = 3$. The states of the Markov chain $V_{G,k}$ are visually ordered in four layers. The first layer consists of the initial state $v_{G,k}$ and a chain of length $n^3 = 27$. The second layer consists of $n = 3$ chains, each of length $3 \cdot n = 9$. The third layer also consists of $n = 3$ chains, each of length $3 \cdot n + 1 = 10$. Finally, the fourth layer consists of a single state t. Edges induced by the probabilistic transition function are indicated by directed lines between states. Chain edges and all other edges of probability 1 are colored in black. Edges of probability $p_1 = \frac{1}{n^3}$ are colored in red, edges of probability $p_2 = \frac{1}{n^{10}}$ in green and edges of probability $p_3 = \frac{1}{n^{50}}$ in blue. The edge of probability $\epsilon - \epsilon_M$ from the initial state $v_{G,k}$ to the first state v^1_1 of the chain in the first layer is colored in purple. Finally, dotted edges depict the remaining edges to the state t in the fourth layer, which are introduced in order to ensure the probabilities of outgoing edges from each state sum up to 1.

Construction of $k_{G,k}$. We set $k_{G,k} := 2 + 3 \cdot n^2 + k$.

Construction of $M_{G,k}$. Fix an enumeration $V = \{v_1, \dots, v_n\}$ of vertices in G. We construct the Markov chain $M_{G,k} = (V_{G,k}, v_{G,k}, \delta_{G,k})$ as follows: The state set $V_{G,k}$ consists of 4 disjoint subsets which we refer to as *layers* (see Figure 1 for a visualization of layers):

- The first layer consists of the initial state $v_{G,k}$ of the Markov chain $M_{G,k}$, as well as of n^3 states $v^1_1, \dots, v^1_{n^3}$ that form a chain of length n^3.
- The second layer consists of n chains where each chain has length $3 \cdot n$. For each $1 \leq i \leq n$ and $1 \leq j \leq 3 \cdot n$, we use $v^2_{i,j}$ to denote the j-th state along the i-th chain.
- The third layer also consists of n chains where each chain has length $3 \cdot n + 1$. For each $1 \leq i \leq n$ and $1 \leq j \leq 3 \cdot n + 1$, we use $v^3_{i,j}$ to denote the j-th state along the i-th chain.
- The fourth layer consists of a single state t.

In addition to transition probabilities defined by the chains specified above, the probabilistic transition relation $\delta_{G,k}$ is defined as follows:

- The last vertex in the chain in the first layer is connected to the vertex t in the fourth layer by an edge of probability 1, i.e. $\delta_{G,k}(v^1_{n^3})(t) = 1$.
- The initial state $v_{G,k}$ is connected to the first state of each chain in the second layer and each chain in the third layer by an edge of probability $p_1 = \frac{1}{n^3}$, i.e. $\delta_{G,k}(v_{G,k})(v^2_{i,1}) = \delta_{G,k}(v_{G,k})(v^3_{i,1}) = \frac{1}{n^3}$ for each $1 \leq i \leq n$.
- For each $1 \leq i \leq n$, the last state in the i-th chain in the second layer is connected to the first states of all but the i-th chain in the third layer by an edge of probability $p_2 = \frac{1}{n^{10}}$, i.e. $\delta_{G,k}(v^2_{i,3\cdot n})(v^3_{j,1}) = \frac{1}{n^{10}}$ for each $j \neq i$.
- For each edge (v_i, v_j) in graph G with $i < j$, the last state of the i-th chain in the second layer and the first state of the j-th chain in the third layer are connected by an edge of probability $p_3 = \frac{1}{n^{50}}$, i.e. $\delta_{G,k}(v^2_{i,3\cdot n})(v^3_{j,1}) = \frac{1}{n^{50}}$.
- The initial state $v_{G,k}$ is connected to the first state v^1_1 of the chain of length n^3 in the first layer by an edge of probability $\epsilon - \epsilon_M$ where $\epsilon_M = n \cdot p_1 + (n-k)\cdot(n-k-1)\cdot p_1 \cdot p_2$, i.e. $\delta_{G,k}(v_{G,k})(v^1_1) = \epsilon - \epsilon_M$. By our choices of p_1 and p_2, one can see that $\epsilon_M \leq \frac{2}{n^2} \leq \frac{1}{n} < \epsilon$ since we assume that $n > \frac{1}{\epsilon} > 1$.
- Vertex t in the fourth layer has a probability 1 self-loop, i.e. $\delta_{G,k}(t)(t) = 1$.
- Finally, for each vertex for which the probabilities of outgoing edges do not sum up to 1, we introduce an additional edge to t of the probability needed to make this sum equal to 1. This ensures that our construction of $M_{G,k}$ indeed yields a Markov chain.

$M_{G,k}$ contains $n^3 + 6 \cdot n^2 + n + 2$ vertices and transition probabilities are of size polynomial in n, hence the size of $M_{G,k}$ is polynomial in the size of G and k.

Correctness of reduction. To prove correctness of the reduction, it remains to show that $(G, k) \in$ VERTEX-COVERif and only if $(M_{G,k}, k_{G,k}) \in \textsc{Core}_\epsilon$.

First, suppose that $(G, k) \in$ VERTEX-COVERand let $K \subseteq V$ be a vertex cover of size k in G. Then, letting

$$C = \{v_{G,k}, t\} \cup \{v^3_{i,1}, \dots, v^2_{i,3n+1} \mid v_i \in K\} \cup \{v^2_{i,1}, \dots, v^3_{i,3n} \mid v_i \notin K\}$$

defines an $(\epsilon, k_{G,k})$-core in $M_{G,k}$. Indeed, we have that $|C| = 2 + k \cdot (3 \cdot n + 1) + (n-k) \cdot 3 \cdot n = 2 + 3 \cdot n^2 + k = k_{G,k}$ and due to our choice of ϵ_M one can verify by inspection that the probability of leaving C in $M_{G,k}$ is exactly equal to ϵ.

Second, we prove that $(M_{G,k}, k_{G,k}) \in \textsc{Core}_\epsilon$ implies $(G, k) \in$ VERTEX-COVERby making a series of observations which will together imply the claim. Define a Markov chain $\bar{M}_{G,k}$ from $M_{G,k}$ by removing the chain $v^1_1, \dots, v^1_{n^3}$ from the first layer and increasing the probability of the transition from the initial state $v_{G,k}$ to the state t in the fourth layer by $\epsilon - \epsilon_M$.

▶ **Observation 2.** *$M_{G,k}$ contains an $(\epsilon, k_{G,k})$-core if and only if $\bar{M}_{G,k}$ contains an $(\epsilon_M, k_{G,k})$-core. Thus, it suffices to show that $(\bar{M}_{G,k}, k_{G,k}) \in \textsc{Core}_{\epsilon_M}$ implies $(G, k) \in$ VERTEX-COVER.*

To see this, note that $n^3 > k_{G,k} = 2 + 3 \cdot n^2 + k$ as we assume that $n \geq 4$ and $k \leq n$. Hence, no $(\epsilon, k_{G,k})$-core in $M_{G,k}$ can contain the whole chain $v^1_1, \dots, v^1_{n^3}$ and the probability of visiting a state in $v^1_1, \dots, v^1_{n^3}$ that is not contained in a $(\epsilon, k_{G,k})$-core is always equal to $\epsilon - \epsilon_M$. Hence, any $(\epsilon, k_{G,k})$-core in $M_{G,k}$ can be modified into an $(\epsilon, k_{G,k})$-core that contains no state in the chain $v_1^1, \dots, v_{n^3}^1$ by removing all states from the chain. The set of states contained in this new core would give rise to an $(\epsilon_M, k_{G,k})$-core in $\bar{M}_{G,k}$. Conversely, a set of states that form an $(\epsilon_M, k_{G,k})$-core in $\bar{M}_{G,k}$ also form an $(\epsilon, k_{G,k})$-core in $M_{G,k}$. This proves Observation 2.

▶ **Observation 3.** *If C is an $(\epsilon_M, k_{G,k})$-core in $\bar{M}_{G,k}$, then removing states of all chains in the second and all chains in the third layer that are not entirely contained in C also gives rise to an $(\epsilon_M, k_{G,k})$-core.*

To see this, observe that each edge between two successive states in a chain has probability 1, therefore if the whole chain is not contained in C then one may remove all states of that chain from C without increasing the probability of a random infinite path in $\bar{M}_{G,k}$ leaving C. Hence, removal of such states from C gives rise to an $(\epsilon_M, k_{G,k})$-core.

▶ **Observation 4.** *If $\bar{M}_{G,k}$ contains an $(\epsilon_M, k_{G,k})$-core, then it also contains an $(\epsilon_M, k_{G,k})$-core that consists of the initial state $v_{G,k}$, the state t and all states of exactly n chains. Furthermore, at most k of these chains are contained in the third layer.*

Consider an $(\epsilon_M, k_{G,k})$-core in $\bar{M}_{G,k}$. The initial state $v_{G,k}$ is contained in the core as otherwise the probability of leaving the core would be $1 > \epsilon_M$. Moreover, a random infinite path in $\bar{M}_{G,k}$ reaches t with probability 1 so as $\epsilon_M < 1$ the state t must also be in the core. Hence, there are at most $k_{G,k} - 2 = 3 \cdot n^2 + k$ remaining states in the core. Now, by Observation 3, we may remove states of chains that are not fully contained in the core to obtain an $(\epsilon_M, k_{G,k})$-core whose remainder consists only of whole chains. Since chains have length $3 \cdot n$ or $3 \cdot n + 1$ and $3 \cdot n \cdot (n+1) > 3 \cdot n^2 + k$, the core must contain states of at most n chains. On the other hand, a random infinite path in $\bar{M}_{G,k}$ leaves the core with probability at least p_1 for every chain not fully contained in the core. So as $(n+1) \cdot p_1 > n \cdot p_1 + (n-k) \cdot (n-k-1) \cdot p_1 \cdot p_2 = \epsilon_M$ due to $p_2 = \frac{1}{n^{10}}$, we may conclude that the core must contain all states from at least n chains as otherwise the probability of leaving the core via an edge of probability p_1 would be at least $(n+1) \cdot p_1$. Thus, the core must consist of $v_{G,k}$, t and all states in exactly n chains. Finally, since the core is of size at most $k_{G,k} = 3 \cdot n^2 + k + 2$ and it must contain $v_{G,k}$ and t, we conclude that at most k of these chains are from the third layer. This proves Observation 4.

▶ **Observation 5.** *If $\bar{M}_{G,k}$ contains an $(\epsilon_M, k_{G,k})$-core, then it also contains an $(\epsilon_M, k_{G,k})$-core such that, for each $1 \leq i \leq n$, the core contains either all states from the i-th chain in the second layer or all states from the i-th chain in the third layer.*

Let C be a set of states in $\bar{M}_{G,k}$ that contains $v_{G,k}$, t and all states of exactly n chains where at most k chains are in the third layer. Observation 4implies that at least one core of such form exists whenever $\bar{M}_{G,k}$ contains an $(\epsilon_M, k_{G,k})$-core. We show that, if C did not satisfy the property in Observation 5, then the probability of leaving C would strictly exceed ϵ_M and thus it would not be an $(\epsilon_M, k_{G,k})$-core in $\bar{M}_{G,k}$. First, by our constructions of $M_{G,k}$ and $\bar{M}_{G,k}$, for each chain in $\bar{M}_{G,k}$ the probability of an edge from the initial state $v_{G,k}$ to the first state in the chain is equal to p_1. Hence, since we assume that C contains exactly n chains and does not contain any states in the remaining n chains, the probability of leaving the core in exactly one step is $n \cdot p_1$. On the other hand, for each $1 \leq i \leq n$, the probability of first entering the i-th chain in the second layer and then moving to a chain in the third layer that is not contained in C is at least:

- $(n-k) \cdot p_1 \cdot p_2$, if the i-th chain in the third layer is also contained in C;
- $(n-k-1) \cdot p_1 \cdot p_2$, otherwise.

This is because C contains at most k chains in the third layer, and the last state in the i-th chain in the second layer is connected to the first states of all but the i-th chains in the third layer by an edge of probability p_2. Therefore, as C contains at least $n-k$ chains in the second layer, we conclude that the probability of leaving C is at least $n \cdot p_1 + (n-k) \cdot (n-k-1) \cdot p_1 \cdot p_2 = \epsilon_M$, with the equality attained if and only if there is no $1 \leq i \leq n$ for which both the i-th chain in the second layer and the i-th chain in the third layer are contained in C. Due to the assumption that C contains all states of exactly n chains, it follows that for each $1 \leq i \leq n$ the set C should contain either the i-th chain in the second layer or the i-th chain in the third layer for the probability of leaving C not to exceed ϵ_M. This concludes the proof of Observation 5.

▶ **Observation 6.** *If $\bar{M}_{G,k}$ contains an $(\epsilon_M, k_{G,k})$-core, then it also contains an $(\epsilon_M, k_{G,k})$-core C for which there does not exist an edge in $\bar{M}_{G,k}$ of probability p_3 whose source state is in C but target state is not in C.*

Suppose that $\bar{M}_{G,k}$ contains an $(\epsilon_M, k_{G,k})$-core. Let C be an $(\epsilon_M, k_{G,k})$-core in $\bar{M}_{G,k}$ which satisfies the properties in Observation 5. It follows from the proof of Observation 5that the probability of a random infinite path in $\bar{M}_{G,k}$ leaving C must be exactly ϵ_M and that this probability is attained solely by taking edges of probabilities p_1 and p_2. Hence, there may not exist an edge in $\bar{M}_{G,k}$ of probability p_3 whose source state is in C but target state is not in C, which proves Observation 6.

▶ **Observation 7.** *If $\bar{M}_{G,k}$ contains an $(\epsilon_M, k_{G,k})$-core then G has a vertex cover of size k. Hence, from Observation 1 we may conclude that $(M_{G,k}, k_{G,k}) \in$ CORE$_\epsilon$ implies $(G,k) \in$ VERTEX-COVER.*

Let C be an $(\epsilon_M, k_{G,k})$-core in $\bar{M}_{G,k}$ that satisfies the properties in the previous observations. We use it to construct a vertex cover of size at most k in G and therefore prove Observation 7. Observations 4and 5imply that C must contain $v_{G,k}$, t and all states of exactly n chains of which at most k are in the third layer, such that for each $1 \leq i \leq n$, C contains either the i-th chain in the second layer or the i-th chain in the third layer. Furthermore, by Observation 6there does not exist an edge in $\bar{M}_{G,k}$ of probability p_3 whose source state is in C but target state is not in C. But, by our construction of $M_{G,k}$ and $\bar{M}_{G,k}$, recall that we have an edge of probability p_3 from the last state in the i-th chain in the second layer to the first state of the j-th chain in the third layer if and only if $i < j$ and (v_i, v_j) is an edge in G. For this not to be an edge whose source state is contained in C but target state is not contained in C, we must either have that the i-th chain in the second layer is not contained in C and therefore the i-th chain in the third layer is contained in C, or that the i-th chain in the second layer and the j-th chain in the third layer are both contained in C. Thus, for each edge (v_i, v_j) in G with $i < j$, the core C should contain at least one of the i-th chain in the third layer or the j-th chain in the third layer. Therefore, as a core must contain at most k chains in the third layer, defining

$$K = \{i \in \{1, \dots, n\} \mid C \text{ contains the } i\text{-th chain in the third layer}\}.$$

gives rise to a vertex cover of size at most k in G. This concludes the proof. ◀

Theorem 1 shows that the problem of deciding whether a Markov chain contains an ϵ-core of at most the given size k is NP-complete. Furthermore, Markov chains $M_{G,k}$ constructed in the proof of Theorem 1 are *acyclic*. This is because edges in $M_{G,k}$ only connect states from lower indexed layers to states in upper indexed layers, and the only cycle contained within a layer is a probability 1 self-loop at the state t in the fourth layer.

In the following theorem, we show that the $\textsc{Core}_\epsilon$ problem remains NP-complete even if we restrict it to acyclic Markov chains that furthermore have maximal vertex degree of at most 3. Hence, even parametrizing Markov chains by the maximal vertex degree would not make the problem solvable in polynomial time, which indicates that the problem of computing cores of at most a given size is computationally a very challenging problem. While Theorem 8 generalizes the result of Theorem 1, the reason why we present them separately is that the construction in the proof of Theorem 8 is more complicated than that in Theorem 1. To that end, we only note that the proof of Theorem 8 modifies the construction of Theorem 1 in a way which bounds the maximal vertex degree of $M_{G,k}$, and we defer the details of this modification to Appendix A.

▶ **Theorem 8** (Proof in Appendix A). *For an $\epsilon \in (0,1)$, let $\textsc{Core}^*_\epsilon$ be the language defined as*

$$\Big\{(M,k) \mid M \text{ is an acyclic Markov chain of maximal} \\ \text{vertex degree } \leq 3 \text{ that contains an } (\epsilon,k)\text{-core}\Big\}.$$

Then, $\textsc{Core}^*_\epsilon$ is NP-complete for any $\epsilon \in (0,1)$.

4 A PTIME Algorithm for Optimal Cores in MCs over CFGs

In this section, we provide a PTIME algorithm that, given a threshold $\epsilon \in (0,1)$ and a Markov chain M over the control-flow graph of a program P, outputs an optimal ϵ-core of M, i.e. an ϵ-core of minimum possible size. As mentioned in Section 1, MCs over CFGs are naturally obtained whenever (statistical) branch prediction is applied to a program. As such, finding a core in such MCs can directly help find a set of lines in the program that covers the vast majority of the runs and is left with very low probability. Any desired static analysis can then be limited to the lines in the core, leading to rigorous probabilistic bounds for the desired property. MCs over CFGs have also been studied in [1].

Before presenting our algorithm, in Section 4.1 we first formally present a grammar for an imperative probabilistic programming language that can define both P and M at the same time. We then present our PTIME algorithm in Section 4.2.

4.1 Markov Chains Induced by Probabilistic Programs

Syntax Grammar. We consider a fragment of finite state first-order imperative probabilistic programs defined by the following grammar:

$$\begin{aligned} \langle prog \rangle = {} & \mathsf{atomic} \\ & \mid \mathsf{if}\ \mathbf{prob}(p)\ \langle prog \rangle\ \mathsf{else}\ \langle prog \rangle \\ & \mid \mathsf{while}\ \mathbf{prob}(p)\ \mathsf{do}\ \langle prog \rangle \\ & \mid \langle prog \rangle\ ;\ \langle prog \rangle \end{aligned} \tag{1}$$

In both the second and the third case, $p \in (0,1)$ is a probability parameter. The first case considers trivial programs that execute a single statement and immediately terminate. The second case considers probabilistic if-branching, where the control-flow follows the if-branch and executes the first program with probability p, and it follows the else-branch and executes the second program with probability $1-p$. The third case considers a construct for probabilistic loops, where, in each iteration, the control-flow enters the loop and executes the inner-nested program with probability p upon which it returns to the loop entry, and it leaves the loop and terminates with probability $1-p$. Finally, the fourth case considers sequential composition of two programs, where the second program is executed upon termination of the first program.

In Markov chains induced by probabilistic programs, in addition to the initial state we also assume the existence of a designated *terminal state*. Thus, for the rest of this section, we slightly modify our definition of Markov chains in Section 2 and define them as tuples $M = (V, s, \delta, t)$, where the last element $t \in V$ denotes the terminal state.

MCs induced by Probabilistic Programs. We now formally define how a probabilistic program generated by the above syntax grammar induces a Markov chain. Given a probabilistic program P, consider its parse tree according to the grammar. In order to define the Markov chain M_P that is induced by P, we start by constructing a Markov chain associated to each leaf node in the parse tree and traverse the parse tree bottom-up in order to construct Markov chains associated to parent programs. The Markov chain M_P associated to the probabilistic program P is then defined to be the Markov chain associated to the root program in the parse tree.

MCs for Leaves. The leaves of the parse tree are trivial atomic statements. A Markov chain associated to an atomic statement is defined via $M_{\mathsf{atomic}} = (V_{\mathsf{atomic}}, s, \delta_{\mathsf{atomic}}, t)$, where

- The state space consists of two states $V_{\mathsf{atomic}} = \{s, t\}$, and
- the probabilistic transition function is defined via $\delta_{\mathsf{atomic}}(s)(t) = \delta_{\mathsf{atomic}}(t)(t) = 1$.

Non-leaf nodes in the parse tree correspond to subprograms that are obtained either by probabilistic if-branching, probabilistic loops or sequential composition constructs. We consider each of the three cases and describe how a Markov chain associated to a parent node program is constructed from Markov chains associated to the children node programs.

Branching Nodes. Suppose that a parent node program is given by

$\mathsf{if}\ \mathbf{prob}(p)\ \mathit{prog}_1\ \mathsf{else}\ \mathit{prog}_2,$

and let $M_1 = (V_1, s_1, \delta_1, t_1)$ and $M_2 = (V_2, s_2, \delta_2, t_2)$ be the Markov chains associated to prog_1 and prog_2, respectively. To construct the Markov chain associated to the parent node program, we introduce two new states s and t and consider $M = (V, s, \delta, t)$, where $V = \{s,t\} \cup V_1 \cup V_2$ and $\delta : V \to \mathcal{D}(V)$ is defined via

- $\delta(s)(s_1) = p$, $\delta(s)(s_2) = 1 - p$,
- $\delta(v) = \delta_1(v)$ for each $v \in V_1 \backslash \{t_1\}$,
- $\delta(v) = \delta_2(v)$ for each $v \in V_2 \backslash \{t_2\}$,
- $\delta(t_1)(t) = \delta(t_2)(t) = 1$, and
- $\delta(t)(t) = 1$.

Intuitively, the Markov chain M has initial state s from which a random infinite path either moves to the initial state s_1 of M_1 with probability p, or to the initial state s_2 of M_2 with probability $1-p$. Then, upon reaching terminal states t_1 of M_1 or t_2 of M_2, a random infinite path moves to the terminal state t of M and stays there indefinitely due to the probability 1 self-loop.

Loop Nodes. Suppose that a parent node program is given by

while $\mathbf{prob}(p)$ do $\mathit{prog_1}$,

and let $M_1 = (V_1, s_1, \delta_1, t_1)$ be the Markov chains associated to prog_1. To construct the Markov chain associated to the parent node program, we introduce two new states s and t and consider $M = (V, s, \delta, t)$, where $V = \{s, t\} \cup V_1$ and $\delta : V \to \mathcal{D}(V)$ is defined via

- $\delta(s)(s_1) = p$, $\delta(s)(t) = 1 - p$,
- $\delta(v) = \delta_1(v)$ for each $v \in V_1 \backslash \{t_1\}$,
- $\delta(t_1)(s) = 1$, and
- $\delta(t)(t) = 1$.

Intuitively, the Markov chain M has initial state s from which a random infinite path either moves to the initial state s_1 of M_1 with probability p, or to the terminal state t of M with probability $1 - p$ where it stays indefinitely due to the probability 1 self-loop. Then, upon reaching terminal states t_1 of M_1, a random infinite path moves to the initial state s of M with probability 1.

Sequential Composition Nodes. Finally, suppose that a parent node program is

prog_1 ; prog_2,

and let $M_1 = (V_1, s_1, \delta_1, t_1)$ and $M_2 = (V_2, s_2, \delta_2, t_2)$ be the Markov chains associated to prog_1 and prog_2, respectively. To construct the Markov chain associated to the parent node program, we introduce two new states s and t and consider $M = (V, s, \delta, t)$, where $V = \{s, t\} \cup V_1 \cup V_2$ and $\delta : V \to \mathcal{D}(V)$ is defined via

- $\delta(s)(s_1) = 1$, $\delta(t_1)(s_2) = 1$, $\delta(t_2)(t) = 1$,
- $\delta(v) = \delta_1(v)$ for each $v \in V_1 \backslash \{t_1\}$,
- $\delta(v) = \delta_2(v)$ for each $v \in V_2 \backslash \{t_2\}$, and
- $\delta(t)(t) = 1$.

Intuitively, the Markov chain M has initial state s from which a random infinite path with probability 1 moves to the initial state s_1 of M_1. Then, upon reaching the terminal state t_1 of M_1, it with probability 1 moves to the initial state s_2 of M_2. Finally, upon reaching the terminal state t_2 of M_2, it with probability 1 moves to the terminal state t of M where it stays indefinitely due to the probability 1 self-loop.

It is easy to see that the Markov chains obtained above are over the CFG of their corresponding program and that, conversely, any MC over a CFG can be obtained by a probabilistic program generated by our grammar.

4.2 PTIME Algorithm for Optimal Core Computation

We now present our polynomial time algorithm for computing the size of a smallest ϵ-core in a Markov chain induced by a probabilistic program generated by the syntax grammar in Equation (1). Given $\epsilon \in [0,1]$ and a probabilistic program P that can be generated by the grammar in Equation (1), our algorithm first constructs its parse tree T_P. It then performs dynamic programming on the parse tree. In particular, it traverses the parse tree bottom-up and for each subprogram P' with associated Markov chain $M_{P'}$ it computes a sequence of non-negative real numbers $(a_{P'}[0], a_{P'}[1], \dots, a_{P'}[|V_{P'}|])$, where $|V_{P'}|$ is the number of states in $M_{P'}$ and

$$a_{P'}[k] = \min\Big\{\epsilon' \geq 0 \mid M_{P'} \text{ contains an } (\epsilon', k)\text{-core}\Big\}$$

for each $0 \leq k \leq |V_{P'}|$. In other words, the k-th entry in the sequence is the minimal probability threshold $\epsilon' \geq 0$ with which a set of k states in the Markov chain $M_{P'}$ may be left. Then, once it has computed the sequence $(a_P[0], \dots, a_P[|V_P|])$ for the root node program P, the algorithm can immediately conclude that the minimal size of an ϵ-core in P is

$$k_{\min} = \min\Big\{1 \leq k \leq |V_P| \mid a_P[k] \leq \epsilon\Big\}.$$

In what follows, we describe how our algorithm computes such a sequence for each program in the parse tree T_P of P. We then prove the correctness of our algorithm and that it runs in polynomial time.

Dynamic Programming on the Parse Tree. The algorithm starts by computing the sequence $(a_{P'}[0], a_{P'}[1], \dots, a_{P'}[|V_{P'}|])$ from each leaf node program P' in the parse tree, upon which it traverses the parse tree bottom-up in order to compute the sequence for each node in the tree. We now describe the dynamic programming steps for each construct type in the syntax grammar:

- *Atomic statement at a leaf node.* Recall that the Markov chain associated to an atomic statement is defined via $M_{\mathsf{atomic}} = (V_{\mathsf{atomic}}, s, \delta_{\mathsf{atomic}}, t)$ with $V_{\mathsf{atomic}} = \{s, t\}$ and $\delta_{\mathsf{atomic}}(s)(t) = \delta_{\mathsf{atomic}}(t)(t) = 1$. Hence, its state space consists of 2 states and its sequence $(a_{\mathsf{atomic}}[0], a_{\mathsf{atomic}}[1], a_{\mathsf{atomic}}[2])$ is defined via

$$a_{\mathsf{atomic}}[k] = \begin{cases} 1, & k = 0,1 \\ 0, & k = 2 \end{cases}$$

  One can easily verify by inspection that each $a_{\mathsf{atomic}}[k]$ is indeed the minimal probability with which a set of k states in M_{atomic} may be left.
- *Probabilistic if-branching node.* Consider now the parse tree node that corresponds to the probabilistic if-branching

$$\mathit{prog} \ = \ \mathsf{if}\ \mathbf{prob}(p)\ \mathit{prog}_1\ \mathsf{else}\ \mathit{prog}_2,$$

  and let $(a_{\mathit{prog}_1}[0], \dots, a_{\mathit{prog}_1}[|V_1|])$ and $(a_{\mathit{prog}_2}[0], \dots, a_{\mathit{prog}_2}[|V_2|])$ be the sequences that the algorithm has computed for the child node programs prog_1 and prog_2 in the parse tree T_P. The algorithm then sets

$$a_{\mathit{prog}}[k] = \begin{cases} 1, & k = 0,1 \\ \min_{j_1, j_2 \geq 0, j_1 + j_2 = k-2} \Big(p \cdot a_{\mathit{prog}_1}[j_1] + (1-p) \cdot a_{\mathit{prog}_2}[j_2]\Big), & k \geq 2 \end{cases}$$

To see that this formula indeed defines the minimal probability with which a set of k states in the Markov chain M_{prog} of *prog* may be left, note that for any positive probability threshold $\epsilon' > 0$, an ϵ'-core in M_{prog} must contain the source state, the terminal state and the total of $k-2$ states in the Markov chains M_{prog_1} of $prog_1$ and M_{prog_2} of $prog_2$. Thus, if we denote by j_1 and j_2 the number of these $k-2$ states that are contained in M_{prog_1} and M_{prog_2}, by the correctness of the algorithm for child node programs we conclude that the above formula minimizes the probability with which a set of k states in the Markov chain M_{prog} of *prog* may be left.

- *Probabilistic loop node.* Next, consider the parse tree node that corresponds to the probabilistic loop

$$prog \;=\; \mathsf{while}\,\mathbf{prob}(p)\,\mathsf{do}\,prog_1\,,$$

and let $(a_{prog_1}[0],\dots,a_{prog_1}[|V_1|])$ be the sequence that the algorithm has computed for the child node program $prog_1$ in the parse tree T_P. The algorithm sets

$$a_{prog}[k] = \begin{cases} 1, & k = 0,1 \\ \frac{p\cdot a_{prog_1}[k-2]}{1-p\cdot(1-a_{prog_1}[k-2])}, & k \geq 2 \end{cases}$$

To see that this formula defines the minimal probability with which a set of k states in the Markov chain M_{prog} of *prog* may be left, note that for any positive probability threshold $\epsilon' > 0$, an ϵ'-core in M_{prog} must contain the source state, the terminal state and the total of $k-2$ states in the Markov chain M_{prog_1} of $prog_1$. The algorithm starts each new loop iteration with probability p, and by the correctness of the algorithm for the root node program we know that in each loop iteration it leaves the set of $k-2$ states in M_{prog_1} with probability at most $a_{prog_1}[k-2]$. Hence, by summing up the geometric series, we conclude that a set of k states in the Markov chain M_{prog} of *prog* may be left with probability at most

$$p\cdot a_{prog_1}[k-2]\cdot\sum_{i=0}^{\infty}\Big(p\cdot(1-a_{prog_1}[k-2])\Big)^i = \frac{p\cdot a_{prog_1}[k-2]}{1-p\cdot(1-a_{prog_1}[k-2])}.$$

- *Sequential decomposition node.* Finally, consider the parse tree node that corresponds to the sequential composition

$$prog \;=\; prog_1\,;\,prog_2,$$

and let $(a_{prog_1}[0],\dots,a_{prog_1}[|V_1|])$ and $(a_{prog_2}[0],\dots,a_{prog_2}[|V_2|])$ be the sequences that the algorithm has computed for the child node programs $prog_1$ and $prog_1$ in the parse tree T_P. The algorithm sets

$$a_{prog}[k] = \begin{cases} 1, & k = 0,1 \\ \min_{j_1,j_2\geq 0, j_1+j_2=k-2}\Big(a_{prog_1}[j_1] + (1-a_{prog_1}[j_1])\cdot a_{prog_2}[j_2]\Big), & k \geq 2 \end{cases}$$

To see that this formula indeed defines the minimal probability with which a set of k states in the Markov chain M_{prog} of *prog* may be left, note that for any positive probability threshold $\epsilon' > 0$, an ϵ'-core in M_{prog} must contain the source state, the terminal state and the total of $k-2$ states in the Markov chains M_{prog_1} of $prog_1$ and M_{prog_2} of $prog_2$. Thus, if we denote by j_1 and j_2 the number of these $k-2$ states that are contained in M_{prog_1} and M_{prog_2}, by the correctness of the algorithm for child node programs we conclude that the above formula minimizes the probability with which a set of k states in the Markov chain M_{prog} of *prog* may be left.

Analysis of each case above and induction on the depth of the parse tree T_P allows us to conclude the following theorem.

▶ **Theorem 9.** *Let P be a probabilistic program generated by the syntax grammar in Equation* (1) *and let M_P be the Markov chain induced by P with state space V_P. Let $(a_P[0], \ldots, a_P[|V_P|])$ be the sequence that the algorithm computes for the program P. Then, for each $0 \leq k \leq |V_P|$, it holds that*

$$a_P[k] = \min\left\{\epsilon' \geq 0 \mid M_P \textit{ contains an } (\epsilon', k)\textit{-core}\right\}.$$

Note that this immediately shows the correctness of our algorithm.

Runtime Analysis. Let $n = |M_P|$ be the size of the Markov chain induced by a probabilistic program P. Note that our dynamic programming algorithm processes each node in the parse tree T_P in $\mathcal{O}(n^2)$ time. Hence, as the size of the parse tree is linear in n, we conclude that the algorithm runs in $\mathcal{O}(n^3)$ time and therefore has polynomial runtime in the size of the underlying Markov chain. Moreover, note that the n in turn is linear in the size of the parse tree which is linear in the size of the program code.

5 Conclusion

An ϵ-core in a Markov chain is a set of states that is left by a random run with probability at most ϵ. In this work, we considered the problem of finding an optimal (smallest) ϵ-core in a given Markov chain M. For every $\epsilon \notin \{0, 1\}$, we proved that the problem is NP-hard. Moreover, this NP-hardness is preserved even when we limit our instances to acyclic MCs with bounded degree 3. Our NP-hardness result answered an open problem posed by [26]. We then showed that for Markov chains over control-flow graphs of structured programs, the problem can be solved in PTIME. In summary, our results demonstrate that there is no efficient algorithm for the general case of the problem unless P=NP, and hence practitioners should use sampling and partial-exploration algorithms with no theoretical guarantees of optimality, such as the one provided by [26]. However, in the important special case of MCs over CFGs, a PTIME algorithm exists.

References

1 Ali Asadi, Krishnendu Chatterjee, Amir Kafshdar Goharshady, Kiarash Mohammadi, and Andreas Pavlogiannis. Faster algorithms for quantitative analysis of MCs and MDPs with small treewidth. In *ATVA*, pages 253–270, 2020.

2 Pranav Ashok, Yuliya Butkova, Holger Hermanns, and Jan Kretínský. Continuous-time Markov decisions based on partial exploration. In *ATVA*, pages 317–334, 2018.

3 Pranav Ashok, Krishnendu Chatterjee, Przemyslaw Daca, Jan Kretínský, and Tobias Meggendorfer. Value iteration for long-run average reward in Markov decision processes. In *CAV*, pages 201–221, 2017.

4 Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, 2008.

5 Kevin Batz, Benjamin Lucien Kaminski, Joost-Pieter Katoen, and Christoph Matheja. Relatively complete verification of probabilistic programs: an expressive language for expectation-based reasoning. In *POPL*, 2021.

6 Tomás Brázdil, Krishnendu Chatterjee, Martin Chmelik, Vojtech Forejt, Jan Kretínský, Marta Z. Kwiatkowska, David Parker, and Mateusz Ujma. Verification of Markov decision processes using learning algorithms. In *ATVA*, pages 98–114, 2014.

7 Krishnendu Chatterjee, Hongfei Fu, and Amir Kafshdar Goharshady. Termination analysis of probabilistic programs through positivstellensatz's. In *CAV*, pages 3–22, 2016.

8 Krishnendu Chatterjee, Hongfei Fu, and Amir Kafshdar Goharshady. Non-polynomial worst-case analysis of recursive programs. In *CAV*, pages 41–63, 2017.

9 Krishnendu Chatterjee, Hongfei Fu, and Amir Kafshdar Goharshady. Non-polynomial worst-case analysis of recursive programs. *ACM Trans. Program. Lang. Syst.*, 41(4):20:1–20:52, 2019.

10 Krishnendu Chatterjee, Hongfei Fu, Amir Kafshdar Goharshady, and Nastaran Okati. Computational approaches for stochastic shortest path on succinct MDPs. In *IJCAI*, pages 4700–4707, 2018.

11 Krishnendu Chatterjee, Amir Kafshdar Goharshady, Rasmus Ibsen-Jensen, and Yaron Velner. Ergodic mean-payoff games for the analysis of attacks in crypto-currencies. In *CONCUR*, pages 11:1–11:17, 2018.

12 Krishnendu Chatterjee, Amir Kafshdar Goharshady, Tobias Meggendorfer, and Đorđe Žikelić. Sound and complete certificates for quantitative termination analysis of probabilistic programs. In *CAV*, 2022.

13 Krishnendu Chatterjee, Amir Kafshdar Goharshady, and Arash Pourdamghani. Probabilistic smart contracts: Secure randomness on the blockchain. In *ICBC*, pages 403–412, 2019.

14 Krishnendu Chatterjee, Amir Kafshdar Goharshady, and Yaron Velner. Quantitative analysis of smart contracts. In *ESOP*, volume 10801, pages 739–767, 2018.

15 Krishnendu Chatterjee, Ehsan Kafshdar Goharshady, Petr Novotný, Jiri Zárevúcky, and Đorđe Žikelić. On lexicographic proof rules for probabilistic termination. In *FM*, volume 13047 of *Lecture Notes in Computer Science*, pages 619–639. Springer, 2021.

16 Krishnendu Chatterjee, Petr Novotný, and Đorđe Žikelić. Stochastic invariants for probabilistic termination. In *POPL*, pages 145–160, 2017.

17 Christian Dehnert, Sebastian Junges, Joost-Pieter Katoen, and Matthias Volk. A storm is coming: A modern probabilistic model checker. In *CAV*, pages 592–600, 2017.

18 Devdatt P. Dubhashi and Alessandro Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms.* Cambridge University Press, 2009.

19 Paul A Gagniuc. *Markov chains: from theory to implementation and experimentation.* Wiley, 2017.

20 Daniel T Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of computational physics*, 22(4):403–434, 1976.

21 S. Gómez, A. Arenas, J. Borge-Holthoefer, S. Meloni, and Y. Moreno. Discrete-time Markov chain approach to contact-based disease spreading in complex networks. *EPL*, 89(3), 2010.

22 Mingzhang Huang, Hongfei Fu, Krishnendu Chatterjee, and Amir Kafshdar Goharshady. Modular verification for almost-sure termination of probabilistic programs. In *OOPSLA*, pages 129:1–129:29, 2019.

23 Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artif. Intell.*, 101(1-2):99–134, 1998.

24 Benjamin Lucien Kaminski, Joost-Pieter Katoen, and Christoph Matheja. On the hardness of analyzing probabilistic programs. *Acta Informatica*, 56(3):255–285, 2019.

25 Hadas Kress-Gazit, Georgios E. Fainekos, and George J. Pappas. Temporal-logic-based reactive mission and motion planning. *IEEE Trans. Robotics*, 25(6):1370–1381, 2009.

26 Jan Kretínský and Tobias Meggendorfer. Of cores: A partial-exploration framework for Markov decision processes. *Log. Methods Comput. Sci.*, 16(4), 2020.

27 Marta Z. Kwiatkowska, Gethin Norman, and David Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *CAV*, pages 585–591, 2011.

28 Annabelle McIver and Carroll Morgan. *Abstraction, Refinement and Proof for Probabilistic Systems.* Monographs in Computer Science. Springer, 2005.

29 Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms.* CRC Press, 1999.

30 Johan Paulsson. Summing up the noise in gene networks. *Nature*, 427(6973):415–418, 2004.

31 Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* Wiley, 1994.

32 Henry Gordon Rice. Classes of recursively enumerable sets and their decision problems. *Transactions of the American Mathematical society*, 74(2):358–366, 1953.

33 James E. Smith. A study of branch prediction strategies. In *ISCA*, pages 202–215, 1998.

34 Jana Wagemaker, Nate Foster, Tobias Kappé, Dexter Kozen, Jurriaan Rot, and Alexandra Silva. Concurrent NetKAT – modeling and analyzing stateful, concurrent networks. In *ESOP*, pages 575–602, 2022.

35 Jinyi Wang, Yican Sun, Hongfei Fu, Krishnendu Chatterjee, and Amir Kafshdar Goharshady. Quantitative analysis of assertion violations in probabilistic programs. In *PLDI*, pages 1171–1186. ACM, 2021.

36 Peixin Wang, Hongfei Fu, Amir Kafshdar Goharshady, Krishnendu Chatterjee, Xudong Qin, and Wenjun Shi. Cost analysis of nondeterministic probabilistic programs. In *PLDI*, pages 204–220. ACM, 2019.

A Proof of Theorem 8

In Theorem 1, we showed that $\textsc{Core}_\epsilon$ is contained in NP. Hence, as acyclic and the maximal vertex degree conditions may be checked in linear time, we conclude that $\textsc{Core}^*_\epsilon$ is also in NP.

To prove that $\textsc{Core}^*_\epsilon$ is NP-hard, we again show a reduction from the $\textsc{Vertex-Cover}$problem. Fix $\epsilon \in (0,1)$ and consider an instance $(G = (V,E), k)$ of the $\textsc{Vertex-Cover}$problem where $n = |V| > \frac{1}{\epsilon}$, $n > 20$ and $n > \frac{1}{\sqrt{1-\epsilon}}$ (these assumptions are required as some inequalities below will only hold for sufficiently large values of n). The reduction is obtained by modifying our construction of $(M_{G,k}, k_{G,k})$ in the proof of Theorem 1 in a way that makes $M_{G,k}$ have the maximal vertex degree of 3 while keeping it acyclic. We denote by $(M^*_{G,k}, k^*_{G,k})$ the newly constructed Markov chain. In what follows, we outline the differences in the construction:

- *Core size.* We set $k^*_{G,k} = 4n^2 + 4n + k$, which is polynomial in the sizes of G and k.
- *Splitting of the initial state.* The initial state $v_{G,k}$ in $M_{G,k}$ is replaced by $2n$ states $v_1^{1,*}, \dots, v_{2n}^{1,*}$, where the first state $v_1^{1,*}$ in the ordering is designated as the initial state of $M^*_{G,k}$. This is done in order to ensure that the maximal vertex degree of each state in the first layer of the Markov chain $M^*_{G,k}$ is at most 3. Note, $v_1^{1,*}, \dots, v_{2n}^{1,*}$ *do not* form a chain as the probabilities of transitions between successive states will not be 1.

  Then, for each $1 \le i \le n$, we set $\delta_{G,k}(v_i^{1,*})(v_{i,1}^2) = p_{1,i}$ and $\delta_{G,k}(v_{n+i}^{1,*})(v_{i,1}^3) = p_{1,n+i}$, where $p_{1,i}$ and $p_{1,n+i}$ are probabilities that are defined inductively as follows. For $i = 1$, we set $p_{1,1} = p_1 = \frac{1}{n^3}$. For $2 \le i \le 2n$, we set

$$p_{1,i} = \frac{p_1}{\prod_{j=1}^{i-1}(1-p_{1,j})}.$$

Then, connect the last state $v_{2n}^{1,*}$ to the first state v_1^1 of the chain $v_1^1, \dots, v_{n^3}^1$ of length n^3 by an edge with probability

$$\delta_{G,k}(v_{2n}^{1,*}, v_1^1) = \frac{\epsilon - \epsilon_M}{\prod_{j=1}^{2n-1}(1-p_{1,j})}$$

so that the probability of reaching the chain of length n^3 from the initial state $v_1^{1,*}$ is equal to $\epsilon - \epsilon_M$ as in the proof of Theorem 1.
Finally, for each $1 \le i < 2n$, we set $\delta_{G,k}(v_i^{1,*})(v_{i+1}^{1,*}) = 1 - p_{1,i}$ and we set $\delta_{G,k}(v_{2n}^{1,*})(t_1^*) = 1 - p_{1,2n} - \frac{\epsilon-\epsilon_M}{\prod_{j=1}^{2n-1}(1-p_{1,j})}$, where t_1^* is the first state in the fourth layer that will be defined in what follows.
To prove that each $p_{1,i} \in [0,1]$, we show by induction on i that $p_{1,i} < \frac{1}{n^2}$. Indeed, $p_{1,1} = \frac{1}{n^3} < \frac{1}{n^2}$ and for $i > 1$ by induction hypothesis we have

$$\begin{aligned} p_{1,i} &= \frac{p_1}{\prod_{j=1}^{i-1}(1-p_{1,j})} \le \frac{p_1}{(1-1/n^2)^{2n}} \le \frac{p_1}{(1-1/n)^{2n}} \\ &= \frac{1/n^3}{(1-1/n)^{2n-2}(1-1/n)^2} < \frac{e^2}{n(n-1)^2} < \frac{1}{n^2}, \end{aligned}$$

for $n \ge 20$, where we use that $(1-1/x)^{-x+1} < e$ for $x > 0$.
The choices of probabilities ensure that a random infinite path in $M_{G,k}^*$ is equally likely to traverse an edge from a state in the first layer to the first state of each chain in the second or the third layer and that this happens with probability p_1 for each chain. Furthermore, this construction ensures that each state in the first layer has a vertex degree of at most 3. Finally, due to the assumption that $n > \frac{1}{\sqrt{1-\epsilon}}$ which is equivalent to $\epsilon < 1 - \frac{1}{n^2}$, we have that the probability of reaching $v_{2n}^{1,*}$ from the initial state $v_1^{1,*}$ is at least

$$\prod_{i=1}^{2n-1}(1-p_{1,i}) > (1-\frac{1}{n^2})^{2n-1} \ge 1 - \frac{1}{n^2} > \epsilon,$$

therefore an ϵ-core in $M_{G,k}$ must contain all states in the sequence $v_1^{1,*}, \dots, v_{2n}^{1,*}$.

- *Extension of sequence lengths.* Sequences in the second layer of $M_{G,k}^*$ have length $4n$, instead of length $3n$ as was the case in our construction of $M_{G,k}$. As in the proof of Theorem 1, for each $1 \le i \le n$ and $1 \le j \le 4n$, we use $v_{i,j}^2$ to denote the j-th state along the i-th sequence. We note that in $M_{G,k}^*$ these sequences *do not* form chains as the probabilities of transitions between successive states will not be 1, as we specify below. Similarly, sequences in the third layer of $M_{G,k}^*$ have length $4n+1$ instead of length $3n+1$ as was the case in $M_{G,k}$ and again they *do not* form chains. For each $1 \le i \le n$ and $1 \le j \le 4n+1$, we use $v_{i,j}^3$ to denote the j-th state along the i-th sequence.
- *Redistributing transition probabilities p_2 in order to bound degrees.* Recall, for each $1 \le i \le n$ and each $j \ne i$, in $M_{G,k}$ we had an edge from $v_{i,3n}^2$ to $v_{j,1}^3$ with $\delta_{G,k}(v_{i,3n}^2)(v_{j,1}^3) = p_2$. We now remove each such edge, and replace it with a new edge from $v_{i,j}^2$ to $v_{j,i}^3$ which has probability $p_{2,i,j}$. The new probabilities are defined as follows. For $i = j$, we set $p_{2,i,i} = 0$. Otherwise, we set $p_{2,i,1} = p_2 = \frac{1}{n^{10}}$ for each $i > 1$ and then for $j \ne i$ and $j > 1$ we inductively define

$$p_{2,i,j} = \frac{p_2}{\prod_{l=1}^{j-1}(1-p_{2,i,l})}.$$

For each $1 \leq i \leq n$, we prove that each $p_{2,i,j} \in [0,1]$ by showing that each $p_{2,i,j} < \frac{1}{n^9}$ by induction on j. Indeed, $p_{2,i,1} = 0$ if $i = 1$ and $p_{2,i,1} = \frac{1}{n^{10}} < \frac{1}{n^9}$ if $i > 1$, as $n \geq 20$. Then, for $j > 1$, by induction hypothesis we have

$$\begin{aligned} p_{2,i,j} &= \frac{p_2}{\prod_{l=1}^{j-1}(1-p_{2,i,l})} \leq \frac{p_2}{(1-1/n)^n} = \frac{1/n^{10}}{(1-1/n)^n} \\ &< \frac{1/n^{10}}{(1-1/n)^{n-1}(1-1/n)} < \frac{e}{n^9(n-1)} < \frac{1}{n^9}. \end{aligned}$$

Finally, for each $v^3_{i,j}$ with $j \leq n$, we set $\delta_{G,k}(v^3_{i,j})(v^3_{i,j+1}) = 1 - p_{2,i,j}$.

This construction ensures that, once a random infinite path in $M_{G,k}$ reaches the first state $v^2_{i,1}$ of the i-th chain in the second layer, it is then equally likely to traverse the sequence $v^2_{i,1}, \ldots, v^2_{i,4n}$ up to the j-th state $v^2_{i,j}$ and then to move to the state $v^3_{j,i}$ of the j-th chain in the third layer and that this happens with probability p_2 for each $1 \leq j \leq n$ with $j \neq i$. Furthermore, it ensures that the vertex degrees due to edges of probability p_2 from the second to the third layer do not exceed 3.

- *Redistributing transition probabilities p_3 in order to bound degrees.* Recall, for each edge (v_i, v_j) in G with $i < j$, in $M_{G,k}$ we had an edge from $v^2_{i,3n}$ to $v^3_{j,1}$ of probability $\delta_{G,k}(v^2_{i,3n})(v^3_{j,1}) = p_3$ where $p_3 = \frac{1}{n^{50}}$.

We now remove this edge and replace them with a new edge from $v^2_{i,n+j}$ to $v^3_{j,n+i}$ of probability $\delta_{G,k}(v^2_{i,n+i})(v^3_{j,n+i}) = p_{3,i,j}$. These probabilities are defined inductively via

$$p_{3,i,j} = \frac{p_3}{\prod_{l=1}^{n}(1-p_{2,i,l})\prod_{l=1}^{j-1}(1-p_{3,i,l})}$$

for each $i < j$ for which (v_i, v_j) is an edge in G, with $p_{3,i,j} = 0$ whenever $i \geq j$ or when (v_i, v_j) is not an edge in G.

For each $1 \leq i \leq n$, we prove that each $p_{3,i,j} \in [0,1]$ by showing that $p_{3,i,j} < \frac{1}{n^{49}}$ by induction on j. Indeed, $p_{3,i,1} = 0$ if (v_i, v_j) is not an edge in G and $p_{3,i,1} = p_3 = \frac{1}{n^{50}} < \frac{1}{n^{49}}$ if (v_i, v_j) is an edge in G as $n \geq 20$. Then, for $j > 1$, by induction hypothesis we have

$$\begin{aligned} p_{3,i,j} &= \frac{p_3}{\prod_{l=1}^{n}(1-p_{2,i,l})\prod_{l=1}^{j-1}(1-p_{3,i,l})} \leq \frac{1/n^{50}}{(1-1/n)^{2n}} \\ &= \frac{1/n^{50}}{(1-1/n)^{2n-2}(1-1/n)^2} < \frac{e^2}{n^{48}(n-1)^2} < \frac{1}{n^{49}}. \end{aligned}$$

Together with the splitting of edges of probability p_2 that we constructed above, this construction ensures that, once a random infinite path in $M_{G,k}$ reaches the first state $v^2_{i,1}$ of the i-th chain in the second layer, it either

- traverses the sequence $v^2_{i,1}, \ldots, v^2_{i,4n}$ up to the j-th state $v^2_{i,j}$ and then to move to the state $v^3_{j,i}$ of the j-th chain in the third layer with probability p_2 for each $j \neq i$, or
- it traverses the sequence $v^2_{i,1}, \ldots, v^2_{i,4n}$ up to the $(n+j)$-th state $v^2_{i,n+j}$ and then to move to the state $v^3_{j,n+i}$ of the j-th chain in the third layer with probability p_3 for each $i < j$ such that (v_i, v_j) is an edge in G, or

  - it traverses the sequence $v^2_{i,1}, \dots, v^2_{i,4n}$ up to the last state $v^2_{i,4n}$ and then moves to the first state t^*_{n+i} in the fourth layer with remaining probability, as specified below.

  Furthermore, the construction ensures that the vertex degrees due to edges of probability p_2 and p_3 from the second to the third layer do not exceed 3.

- The single state t in the fourth layer of $M_{G,k}$ is replaced by a chain of $2n$ states $t^*_1, \dots, t^*_{2n}$, with a probability 1 self-loop at t^*_{2n}. For each $1 \leq i \leq n$, the last state $v^3_{i,4n+1}$ of the i-th chain in the third layer is connected to t^*_i by an edge of probability 1. For each $1 \leq i \leq n$, the last state $v^2_{i,4n}$ of the i-th chain in the second layer is connected to t^*_{n+i} by an edge of probability 1.

Note that $M^*_{G,k}$ contains n^3+8n^2+5n states and transition probabilities are of size polynomial in n, therefore $M^*_{G,k}$ is polynomial in the size of G and k.

Correctness of reduction. While our modified construction ensures that the maximal vertex degree in the constructed Markov chain $M^*_{G,k}$ is equal to 3 and that $M^*_{G,k}$ remains acyclic, it preserves the key properties of $M_{G,k}$ about probabilities of moving between different chains. Therefore, one can follow the sequence of observations in the proof of Theorem 1 and analogously prove correctness of this modified reduction. Due to the proofs being analogous, we omit the details.