

Optimal Stopping of Self-Refining Foundation Models

Kim Hammar, Tansu Alpcan, and Emil C. Lupu

Abstract—Foundation models can improve their outputs through a self-refinement process driven by external feedback. In this process, the model is embedded in an iterative loop where it generates outputs, receives feedback from verifiers, and refines its responses through in-context learning. Following a novel approach, we formalize this process as an optimal stopping problem where the number of refinement iterations is decided based on expected improvement relative to cost. We derive optimal stopping policies and show that they can be efficiently computed through stochastic approximation. To evaluate our approach experimentally, we apply it to a coding benchmark for foundation models. The empirical results show that our stopping policies are significantly more cost-efficient than stopping policies proposed in prior work.

I. INTRODUCTION

Foundation models are becoming an important component of decision-making systems across many domains, including software development [1], scientific research [2], content generation [3], and systems engineering [4]. They consist of large neural networks that are trained on vast datasets (e.g., web-scale text corpora), which enables them to generalize across tasks and modalities. Prominent examples include large language models (e.g., Gemini [5]), time-series models (e.g., Chronos [6]), and visual models (e.g., Flamingo [7]).

These types of models generate outputs autoregressively by sampling from a conditional distribution $p_{\theta}(z | v)$, where θ denotes the model parameters, v is an input (e.g., a task description), and z is the generated output (e.g., code or text). Since the model adapts to the input v , the same model can be applied to different tasks without updating the parameters θ , a property known as *in-context learning* [8]. This flexibility also enables the model to refine its outputs based on feedback. In particular, given an initial output z and feedback x (e.g., from automated verification procedures), the model can generate a revised output by sampling from $p_{\theta}(z' | v, z, x)$, as initially shown by Madaan et al. [9].

The process of generating and revising outputs based on feedback typically continues for a fixed number of iterations or until a predetermined stopping criterion is met. While prior work has demonstrated that this approach improves outputs across various application domains, such as coding (see e.g., [10]) and logical reasoning (see e.g., [11], [12]), the literature is focused on empirical evaluation and lacks a formal analysis. In particular, there is little understanding of how to optimally decide when to stop refining.

K. Hammar is supported by the Swedish Research Council under contract 2024-06436.

T. Alpcan is with the University of Melbourne, Australia. tansu.alpcan@unimelb.edu.au.

K. Hammar and E.C. Lupu are with Imperial College London, United Kingdom. {k.hammar,e.c.lupu}@imperial.ac.uk.

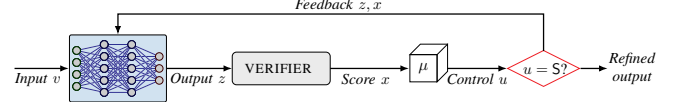


Fig. 1. Illustration of a self-refining foundation model with stopping. The model is integrated into an iterative verification and refinement loop, in which it is used to generate outputs that are verified against constraints to generate a score. The score is then input to a stopping policy μ that decides whether to stop and accept the output or to continue revising it.

In this paper, we address this research gap by presenting a decision-theoretic model of the self-refinement process. Specifically, we formulate self-refinement as an optimal stopping problem in which the number of refinement iterations is decided based on the expected improvement relative to the cost of invoking the foundation model. For example, if the foundation model is hosted locally, each invocation incurs computational expenditure. Similarly, if the model is accessed via an external provider, it incurs a monetary cost.

Leveraging this mathematical formulation, we establish conditions for optimal threshold-based stopping policies and validate them on a coding benchmark across three frontier models. Empirical evaluations show that our stopping policies are cost-efficient and consistently improve performance when compared to other approaches proposed in the research literature, which rely on heuristic stopping policies.

Our contributions can be summarized as follows:

- We present a novel formulation of self-refinement in foundation models as an optimal stopping problem.
- We derive optimal stopping policies and validate them on a coding benchmark. The results show that they are more cost-efficient than stopping policies proposed in prior work. Our implementation is available at [13].

II. RELATED WORK

Optimal stopping is a classical problem with a well-developed theory; see e.g., Wald [14], Shiryaev [15], and Chow et al. [16]. Example use cases for this theory include asset selling [17], intrusion detection [18], network management [19], machine replacement [20], hypothesis testing [14], gambling [16], industrial control [21], change detection [22], [23], [24], [25], [26], and flow control [27], [28], [29]. To our knowledge, we are the first to apply this theory to analyze the self-refinement process of a foundation model.

Many variants of the stopping problem have been studied. Examples include discrete-time and continuous-time problems, finite-horizon and infinite-horizon problems, problems with fully observed and partially observed state spaces, problems with finite and infinite state spaces, Markovian

and non-Markovian problems, and single-stop and multi-stop problems. Consequently, different solution approaches have been developed. The most common are the *martingale* approach (see e.g., Snell [30]) and the *Markovian* approach (see e.g., Bather [31]). In this paper, we investigate a stopping problem with a finite time horizon, discrete-time progression, a continuous state space, and the Markov property.

III. THE SELF-REFINEMENT USE CASE

We consider a scenario where a foundation model is used to solve a task specified by an instruction in natural language. As an example, the task could be to solve a programming problem and the model could be a large language model, such as Gemini [5]. To evaluate the solution produced by the model, we associate it with a *score* $x \in [0, 1]$, with $x = 1$ being the optimal score. For instance, in the programming example, the solution could be a piece of code and the score x could capture its correctness and computational efficiency.

We assume that the model is embedded in an iterative loop where it refines its output based on feedback, as illustrated in Fig. 1. Each iteration invokes the model with the current score $x \in [0, 1]$ as feedback, which leads to a new output that receives an updated score x' and incurs a cost $c > 0$. This cost can be either computational (if the model is run locally) or monetary (if the model is provided externally).

After each refinement iteration, a stopping policy μ uses the current score to decide whether to stop and accept the output or to continue refining, where the maximum number of iterations is N . When designing this policy, the goal is to optimally balance the refinement costs against the potential improvement in the score x , as formally defined below.

IV. OPTIMAL STOPPING FORMULATION

We formalize the self-refinement use case described above as a discrete-time dynamical system where the score evolves as a Markov process $(x_k)_{k=0}^N$. At each stage $k \in \{0, 1, \dots, N-1\}$, two controls are available: (S)top and (C)ontinue. Control $u = S$ in state x yields a payoff $g(x) \geq 0$ that quantifies the quality of the output and terminates the process. Conversely, control $u = C$ incurs a cost $c > 0$ and transitions the system to the next state according to

$$x_{k+1} = f(x_k, w_k), \quad k = 0, 1, \dots, N-1, \quad (1)$$

where the initial state x_0 is given, $(w_k)_{k=0}^{N-1}$ are i.i.d. random variables with distribution P_w that capture the stochasticity of the generation process, and f is the system function.

Let $(\Omega, \mathcal{F}, P)$ be the underlying probability space and let $(\mathcal{F}_k)_{k=0}^N$ denote the filtration generated by the quality scores, i.e., each $\mathcal{F}_k$ is the sigma-algebra generated by the random variables $x_0, \dots, x_k$. Within this space, we define a stopping policy $\mu = (\mu_0, \mu_1, \dots, \mu_N)$ as a sequence of measurable functions $\mu_k : [0, 1] \rightarrow \{S, C\}$ that map the state to a control.

Given a policy μ , we define the stopping time as

$$\tau_\mu = \min\{k \in \{0, 1, \dots, N\} \mid \mu_k(x_k) = S\}, \quad (2)$$

with the constraint $\mu_N(x_N) = S$, which ensures $\tau_\mu \leq N$.

Proposition 1. The random variable τ_μ defined in (2) is a stopping time with respect to the filtration $(\mathcal{F}_k)_{k=0}^N$.

Proof. It suffices to show that the event $\{\tau_\mu = k\}$ belongs to the sigma-algebra $\mathcal{F}_k$ for all $k \in \{0, 1, \dots, N\}$. For $k < N$, $\{\tau_\mu = k\}$ is the event that the stopping policy continues at all preceding iterations and stops at iteration k , i.e.,

$$\{\tau_\mu = k\} = \left(\bigcap_{j=0}^{k-1} \{\mu_j(x_j) = C\} \right) \cap \{\mu_k(x_k) = S\},$$

where the intersection is Ω when $k = 0$.

Since each policy μ_j is a measurable function of x_j and x_j is $\mathcal{F}_j$ -measurable (and hence $\mathcal{F}_k$ -measurable for $j \leq k$), each set in the intersection belongs to $\mathcal{F}_k$. As a consequence, we have $\{\tau_\mu = k\} \in \mathcal{F}_k$. For $k = N$, the event $\{\tau_\mu = N\} = \bigcap_{j=0}^{N-1} \{\mu_j(x_j) = C\}$ belongs to $\mathcal{F}_{N-1} \subseteq \mathcal{F}_N$. Thus τ_μ is a stopping time with respect to $(\mathcal{F}_k)_{k=0}^N$. $\square$

Given the preceding definition of a stopping policy μ , we define the stopping and continuation sets as

$$\mathcal{S}_k^\mu = \{x \mid x \in [0, 1], \mu_k(x) = S\}, \quad (\text{Stopping set})$$

$$\mathcal{C}_k^\mu = \{x \mid x \in [0, 1], \mu_k(x) = C\}. \quad (\text{Continuation set})$$

When designing the stopping policy μ , the objective is to maximize the expected payoff $E\{g(x_{\tau_\mu})\}$ while minimizing refinement costs. In particular, each refinement iteration incurs a cost $c > 0$ (e.g., per-token charges) and the objective is to find an optimal stopping policy μ^* that satisfies

$$\mu^* \in \arg \max_{\mu} E\{g(x_{\tau_\mu}) - c\tau_\mu\} \quad \text{subject to (1)}, \quad (3)$$

where the expectation is taken over the randomness in the dynamics (1) and the stopping time τ_μ is defined in (2). We denote the optimal stopping time, stopping set, and continuation set by τ^* , $\mathcal{S}_k^*$, and $\mathcal{C}_k^*$, respectively.

Remark 1. We model the cost c as a constant. In principle, c could vary with the stage k , e.g., if token consumption changes across iterations. However, in our experimental evaluation (see §V), token consumption per iteration is approximately constant, which justifies this assumption.

Characterizing the Optimal Stopping Policy

To characterize the optimal stopping policy for problem (3), we use a dynamic programming formulation. Following this approach, we define the stage-dependent stopping time

$$\tau_{\mu,k} = \min\{j \in \{k, k+1, \dots, N\} \mid \mu_j(x_j) = S\},$$

so that $\tau_\mu = \tau_{\mu,0}$; cf. (2). We then define the value function $V_k^\mu : [0, 1] \rightarrow \mathbb{R}$ to represent the expected payoff minus refinement costs from state x at stage k under policy μ , i.e.,

$$V_k^\mu(x) = E\{g(x_{\tau_{\mu,k}}) - c(\tau_{\mu,k} - k) \mid x_k = x\},$$

for all stages $k = 0, 1, \dots, N-1$ and states $x \in [0, 1]$, with the terminal condition $V_N^\mu(x) = g(x)$ for all $x \in [0, 1]$.

The optimal value function V_k^* , which is derived by optimizing over all policies μ , satisfies the Bellman equation

$$V_k^*(x) = \max \left\{ g(x), -c + E\{V_{k+1}^*(f(x, w_k))\} \right\}, \quad (4)$$

for all stages $k = 0, 1, \dots, N - 1$ and states $x \in [0, 1]$.

The first term in the maximization (4) corresponds to the decision to stop ($u = S$), which yields the payoff $g(x)$. The second term corresponds to the decision to continue ($u = C$), which incurs the cost c and leads to a new state. We denote the value of continuing under an optimal policy by

$$Q_k^*(x) = -c + E\{V_{k+1}^*(f(x, w_k))\}.$$

Given this notation, the optimal policy can be expressed as

$$\mu_k^*(x) = \begin{cases} S, & \text{if } g(x) \geq Q_k^*(x), \\ C, & \text{otherwise,} \end{cases} \quad k = 0, 1, \dots, N - 1.$$

The above structure implies that the sequence $(V_k^*)_{k=0}^N$ is the smallest superharmonic majorant of g .¹ In other words, the stopping set is the set of states where V_k^* equals g , while the continuation set is the set where V_k^* exceeds g . This characterization admits a geometric interpretation of the optimal stopping policy, as illustrated in Fig. 2.

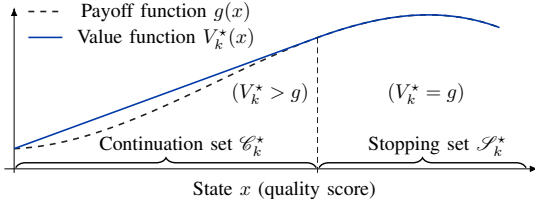


Fig. 2. Geometric characterization of the optimal stopping policy.

Remark 2. When $c = 0$, the Bellman equation (4) reduces to $V_k^*(x) = \max\{g(x), E\{V_{k+1}^*(f(x, w_k))\}\}$. In this case, the process $(V_k^*(x_k))_{k=0}^N$ is the Snell envelope of the payoff process $(g(x_k))_{k=0}^N$ [30] and the problem corresponds to the classical optimal stopping formulation of Dynkin [32].

V. IDENTIFICATION OF THE STOPPING PROBLEM

The optimal stopping problem formulated in the preceding section is defined by three components: the transition dynamics f , the continuation cost c , and the payoff function g ; cf. (3). In this section, we identify these components based on empirical observations of three foundation models.

A. Example Use Case: Code Optimization

The optimal stopping formulation presented in (3) is general and applies to any task that a foundation model can be used to solve, provided that the output of the model can be associated with a quality score. However, for the purpose of analysis and experimental validation, we instantiate the stopping problem for a specific *code optimization use case*.

In this use case, the input to the foundation model is a programming task and the output is code for solving the task. At each refinement iteration, the generated code is executed against a test suite that measures its correctness,

¹A sequence $(h_k)_{k=0}^N$ is *superharmonic* with respect to the system $f(x, w)$ if $h_k(x) \geq -c + E\{h_{k+1}(f(x, w))\}$ for all x and $k < N$. It is a *superharmonic majorant* of g if additionally $h_k(x) \geq g(x)$ everywhere.

execution time, and memory usage. These measurements are aggregated into a candidate score $\tilde{x}_k \in [0, 1]$ defined as

$$\tilde{x}_k = \begin{cases} 0, & \text{if tests fail,} \\ \text{clip}\left(\frac{T_{\max} - T_k}{T_{\max} - T_{\min}}, 0, 1\right), & \text{otherwise,} \end{cases} \quad (5a)$$

$$T_k = \frac{1}{T_{\text{ref}}} \int_0^{\mathcal{T}_k} m_k(t) dt, \quad (5b)$$

where $m_k(t)$ is the memory footprint at time t when executing the code generated at iteration k , $\mathcal{T}_k$ is the total execution time, $\text{clip}(\cdot, 0, 1)$ restricts the input to the unit interval, and T_{ref} is the memory-over-time integral of a given reference code, as defined in [33]. The ratio T_k thus captures both execution time and memory consumption in a single metric. The constants $T_{\min}$ and $T_{\max}$ are normalization bounds chosen so that code that passes the tests with $T_k \leq T_{\min}$ receives the maximal score $\tilde{x}_k = 1$, while failing code and code that passes the tests with $T_k \geq T_{\max}$ receive $\tilde{x}_k = 0$.

After computing the candidate score $\tilde{x}_k$ through (5), we update the state of the stopping problem, which is the score of the best code retained so far, i.e., $x_k = \max\{x_{k-1}, \tilde{x}_k\}$ with initial state $x_0 = \tilde{x}_0$. Next, we feed the updated state x_k to the stopping policy μ , which decides whether to stop or continue the refinement process by generating new code.

B. Data Collection

We collect data using EFFIBENCH [33], which is a coding benchmark for large language models. This benchmark consists of programming tasks for which solutions are assessed based on the candidate score defined in (5). We apply three frontier foundation models to this benchmark: HAIKU 4.5 [34], GEMINI FLASH-LITE 3.1 [5], and GPT CODEX MINI 5.1 [35]. For each model, we run the self-refinement loop illustrated in Fig. 1 for $N = 10$ iterations across 50 tasks.

Figure 3 (on the next page) summarizes the collected data. The top row shows that all three models improve code efficiency through self-refinement, but with diminishing returns. Most of the improvement occurs within the first 3 – 4 iterations. Moreover, the middle and bottom rows in Fig. 3 show that token consumption and monetary cost grow approximately linearly with the number of refinement iterations. Together, these trends illustrate the trade-off between the quality of the generated output and the cost of refinement.

C. System Identification

We use the collected data to estimate the parameters of the stopping problem as follows. First, we define the continuation cost c [cf. (3)] to be the average monetary cost of performing a self-refinement iteration. This cost depends on the choice of foundation model and is defined in Table I.

Foundation model	Continuation cost c	Explanation
HAIKU 4.5	0.01	Avg. cost (\$) per iteration; see Fig. 3.
GEMINI FLASH-LITE 3.1	0.0025	Avg. cost (\$) per iteration; see Fig. 3.
GPT CODEX MINI 5.1	0.005	Avg. cost (\$) per iteration; see Fig. 3.

TABLE I

THE CONTINUATION COST FOR THE THREE FOUNDATION MODELS.
(COMPUTED BASED ON PER-TOKEN CHARGES AS OF MARCH 17, 2026.)

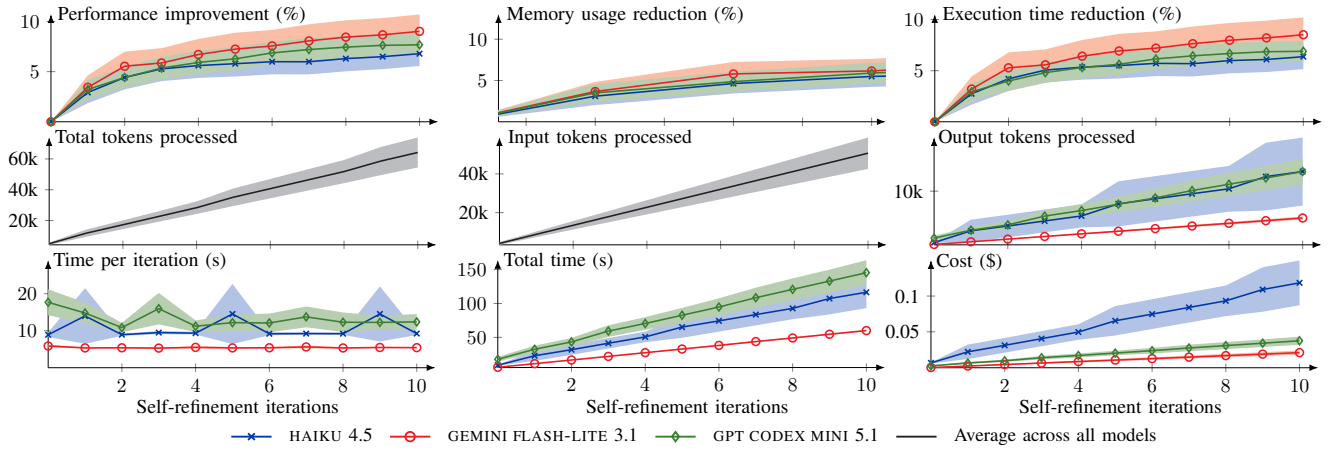


Fig. 3. Empirical measurements of the self-refinement process across three frontier models on EFFIBENCH [33]. The top row shows performance metrics (execution time and memory usage) as functions of the refinement iteration. The middle row shows cumulative token consumption (total, input, and output). The bottom row shows the refinement time (per iteration and cumulative) and cumulative monetary cost. Curves show the mean values and shaded regions indicate one standard deviation. The cost is computed from per-token charges as listed by each model provider on March 17, 2026.

Second, we define the payoff function g as $g(x) = \beta x$, where $\beta > 0$ is a weighting factor that allows controlling the relative importance of output quality compared to refinement cost. This parameter is not estimated from data; rather, it is a configuration parameter that can be adapted to the use case.

Lastly, we estimate the system function f [cf. (1)] from empirical observations of the refinement process. In particular, we model the transition dynamics as a regression problem where the candidate score $\tilde{x}_{k+1}$ [cf. (5)] is predicted from the current state x_k ; the state transition then follows from $x_{k+1} = \max\{x_k, \tilde{x}_{k+1}\}$. To capture both the nonlinear relationship and the inherent uncertainty in the refinement process, we estimate f using a Gaussian process (GP) model.

Formally, we assume that f can be represented as

$$x_{k+1} = f(x_k, w_k) = \min\{1, \max\{x_k, q(x_k) + w_k\}\}, \quad (6)$$

where $k = 0, 1, \dots, N-1$ is the stage of the refinement process, q is an unknown *score function*, and w_k is a zero-mean Gaussian noise variable with variance σ^2 . The max operator in (6) reflects the fact that the previous code is retained if the newly generated code achieves a lower score. Similarly, the min operator is used to ensure that $x \in [0, 1]$.

To estimate the unknown function q , we use a Gaussian process prior, which allows us to obtain a posterior distribution over functions conditioned on the observed data. Specifically, we define the prior as $q \sim \mathcal{GP}(m, \kappa)$, where m is the mean function and κ is the covariance function of the GP. In this paper, we define these functions as

$$m(x) = x, \quad (7a)$$

$$\kappa(x, x') = \left(1 + \sqrt{5}r + \frac{5r^2}{3}\right) \exp(-\sqrt{5}r), \quad (7b)$$

for all states $x, x' \in [0, 1]$, where $r = |x - x'|$. This covariance function encodes the assumption that q varies smoothly over the input space. Similarly, the mean function encodes that refinement preserves the current quality score.

Given this prior, we update it using the empirical observations from evaluations on EFFIBENCH [33] (see Fig. 3) and Bayes' rule. (We set σ^2 equal to the variance estimated during the GP fit.)² We denote the mean of the resulting posterior as $\tilde{q}(x) = E\{q(x) \mid \text{observed data}\}$. Figure 4 shows the posteriors for the three foundation models. We observe that the model $\tilde{q}(x)$ exhibits diminishing returns as the state x increases. In particular, for lower-quality scores (states), the gain from refinement is relatively large, while for higher-quality scores, the improvement becomes smaller.

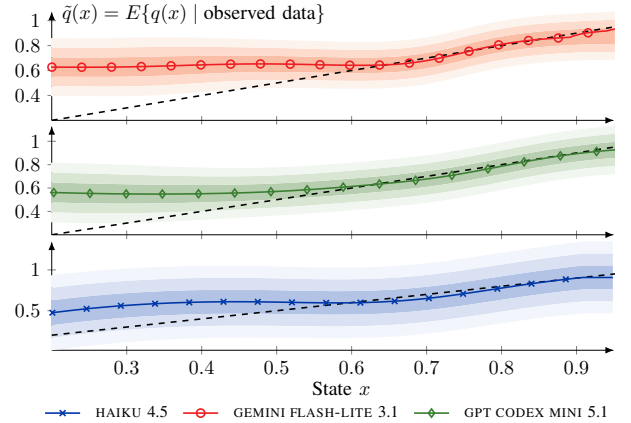


Fig. 4. Estimation of the score function q [cf. (6)] when applying the foundation models to EFFIBENCH [33]. Curves show the mean values of the Gaussian processes; shaded regions indicate one, two, and three standard deviations (darker to lighter shades). The dashed lines show $\tilde{q}(x) = x$.

From the identified models in Fig. 4, we extract the following two structural observations of the system dynamics.

²See Rasmussen and Williams for detailed formulas of the Bayesian updates [36, Def. 2.1]. Due to clipping [cf. (5)], a candidate score $\tilde{x}_{k+1} \in \{0, 1\}$ may correspond to a value of $q(x_k) + w_k$ outside $[0, 1]$. We nevertheless treat all candidate scores as direct observations of $q(x_k) + w_k$, which yields a Gaussian approximation of the posterior.

Observation 1 (Monotonicity). *The identified model $\tilde{q}$ [cf. (6)] is nondecreasing on $[0, 1]$; see Fig. 4. Since the pointwise minimum and maximum of two nondecreasing functions are also nondecreasing, it follows that $f(x, w) = \min\{1, \max\{x, \tilde{q}(x) + w\}\}$ is nondecreasing in x .*

Observation 2 (Diminishing returns). *The difference $\tilde{q}(x) - x$ [cf. (6)] is nonincreasing on $[0, 1]$; see Fig. 4. Since the operations $\min\{1 - x, \dots\}$ and $\max\{0, \dots\}$ preserve this nonincreasing structure, it follows that $f(x, w) - x = \min\{1 - x, \max\{0, \tilde{q}(x) - x + w\}\}$ is nonincreasing in x .*

These observations will be exploited in the next section to derive theoretical properties of an optimal stopping policy.

VI. STRUCTURE OF AN OPTIMAL STOPPING POLICY

Leveraging the identified system dynamics (see Fig. 4), we now derive structural properties of an optimal stopping policy. In the following analysis, we treat Observations 1–2 as exact characteristics of the system function f [cf. (1)] and assume the linear payoff structure defined in §V-C. That is, we model the GP posterior mean $\tilde{q}$ (which is continuous) as the score function q in (6) and define the payoff function g as $g(x) = \beta x$, where $\beta > 0$ is a weighting factor.

Lemma 1. For each stage $k \in \{0, 1, \dots, N\}$,

- 1) The value function $V_k^*(x)$ is nondecreasing in x .
- 2) The difference $V_k^*(x) - g(x)$ is nonincreasing in x .
- 3) The functions V_k^* and Q_k^* (for $k < N$) are continuous.

Proof. We proceed by induction on k .

Base case: $V_N^*(x) = g(x) = \beta x$ is nondecreasing and continuous, and $V_N^*(x) - g(x) = 0$ is nonincreasing.

Inductive step: Assume that V_{k+1}^* is nondecreasing and continuous, and that $V_{k+1}^*(y) - g(y)$ is nonincreasing in y . Since $f(x, w)$ is nondecreasing in x by Observation 1 and V_{k+1}^* is nondecreasing by the inductive hypothesis, $Q_k^*(x) = -c + E\{V_{k+1}^*(f(x, w_k))\}$ is nondecreasing in x . Since $g(x)$ is also nondecreasing, $V_k^*(x) = \max\{g(x), Q_k^*(x)\}$ is nondecreasing. This completes the proof of the first property.

For the second property, we start by expanding $V_k^*(x) - g(x)$ as $V_k^*(x) - g(x) = \max\{0, Q_k^*(x) - g(x)\}$, which implies that it suffices to show that $Q_k^*(x) - g(x)$ is nonincreasing. We expand this expression as

$$Q_k^*(x) - g(x) = -c + E\{V_{k+1}^*(f(x, w_k)) - g(f(x, w_k))\} + \beta E\{f(x, w_k) - x\}.$$

By the inductive hypothesis, $V_{k+1}^*(y) - g(y)$ is nonincreasing in y . Since $f(x, w)$ is nondecreasing in x by Observation 1, the first expectation is nonincreasing in x . By Observation 2, $f(x, w_k) - x$ is nonincreasing in x , and so is the second expectation. As the sum of nonincreasing functions is nonincreasing, $Q_k^*(x) - g(x)$ is nonincreasing in x .

For the third property, $f(x, w) = \min\{1, \max\{x, q(x) + w\}\}$ is continuous in x for each w since q is continuous. Hence, since V_{k+1}^* is continuous by the inductive hypothesis, $x \mapsto V_{k+1}^*(f(x, w))$ is continuous and bounded for each w . As a consequence, $Q_k^*(x) = -c + E\{V_{k+1}^*(f(x, w_k))\}$ is

continuous by the dominated convergence theorem. Therefore, $V_k^* = \max\{g, Q_k^*\}$ is the maximum of two continuous functions, which implies that V_k^* is continuous. $\square$

Given the above lemma, there exists an optimal policy with threshold structure, as formally stated below.

Proposition 2. There exist an optimal policy μ^* and thresholds $\alpha_0, \alpha_1, \dots, \alpha_{N-1} \in [0, 1]$ such that

$$\mu_k^*(x) = \begin{cases} S & \text{if } x \geq \alpha_k, \\ C & \text{if } x < \alpha_k, \end{cases} \quad k = 0, 1, \dots, N-1. \quad (8)$$

Proof. The optimal policy stops at stage k if and only if $g(x) \geq Q_k^*(x)$. By Lemma 1, $Q_k^*(x) - g(x)$ is nonincreasing in x . Therefore, if $g(x') \geq Q_k^*(x')$ for some $x' \in [0, 1]$, then $g(x) \geq Q_k^*(x)$ for all $x \geq x'$. This means that the stopping set $\mathcal{S}_k^* = \{x \mid x \in [0, 1], g(x) \geq Q_k^*(x)\}$ is an upper set, i.e., if $x \in \mathcal{S}_k^*$ then $x' \in \mathcal{S}_k^*$ for all $x' \in [x, 1]$. It remains to show that $\mathcal{S}_k^*$ is nonempty and closed. For nonemptiness, note that $f(1, w) = 1$ for all w [cf. (6)]. Hence, any policy that continues $j \geq 0$ times from state 1 receives $g(1) - jc$, which is maximized at $j = 0$. Therefore $V_{k+1}^*(1) = g(1)$, which gives $Q_k^*(1) = g(1) - c < g(1)$ and thus $1 \in \mathcal{S}_k^*$. Further, $\mathcal{S}_k^*$ is closed since $g - Q_k^*$ is continuous by Lemma 1. A nonempty closed upper subset of $[0, 1]$ has the form $[\alpha_k, 1]$ with $\alpha_k = \min \mathcal{S}_k^*$, which yields (8). $\square$

Proposition 3. The optimal thresholds [cf. (8)] are nonincreasing in the stage k , i.e., $\alpha_0 \geq \alpha_1 \geq \dots \geq \alpha_{N-1}$.

Proof. We start by showing that $V_k^*(x) \geq V_{k+1}^*(x)$ for all stages $k \in \{0, 1, \dots, N-1\}$ and states $x \in [0, 1]$.

Base case ($k = N-1$): We have $V_{N-1}^*(x) = \max\{g(x), -c + E\{g(f(x, w))\}\} \geq g(x) = V_N^*(x)$.

Inductive step: Assume $V_{k+1}^*(x) \geq V_{k+2}^*(x)$ for all states $x \in [0, 1]$. Then

$$\begin{aligned} Q_k^*(x) &= -c + E\{V_{k+1}^*(f(x, w))\} \\ &\geq -c + E\{V_{k+2}^*(f(x, w))\} \\ &= Q_{k+1}^*(x). \end{aligned}$$

Consequently,

$$\begin{aligned} V_k^*(x) &= \max\{g(x), Q_k^*(x)\} \\ &\geq \max\{g(x), Q_{k+1}^*(x)\} \\ &= V_{k+1}^*(x). \end{aligned}$$

Since $Q_k^*(x) \geq Q_{k+1}^*(x)$, we have

$$\begin{aligned} \mathcal{S}_k^* &= \{x \mid x \in [0, 1], g(x) \geq Q_k^*(x)\} \\ &\subseteq \{x \in [0, 1] \mid g(x) \geq Q_{k+1}^*(x)\} = \mathcal{S}_{k+1}^*. \end{aligned}$$

Taking minima of $\mathcal{S}_k^*$ yields $\alpha_k \geq \alpha_{k+1}$. $\square$

The preceding propositions imply that each stopping set $\mathcal{S}_k^*$ has the form $[\alpha_k, 1]$ for some threshold $\alpha_k \in [0, 1]$. We now show that these sets are all equal.

Proposition 4. The optimal stopping sets satisfy $\mathcal{S}_0^* = \mathcal{S}_1^* = \dots = \mathcal{S}_{N-1}^*$. Equivalently, the optimal thresholds are stage-independent, i.e., $\alpha_0 = \alpha_1 = \dots = \alpha_{N-1} = \alpha^*$.

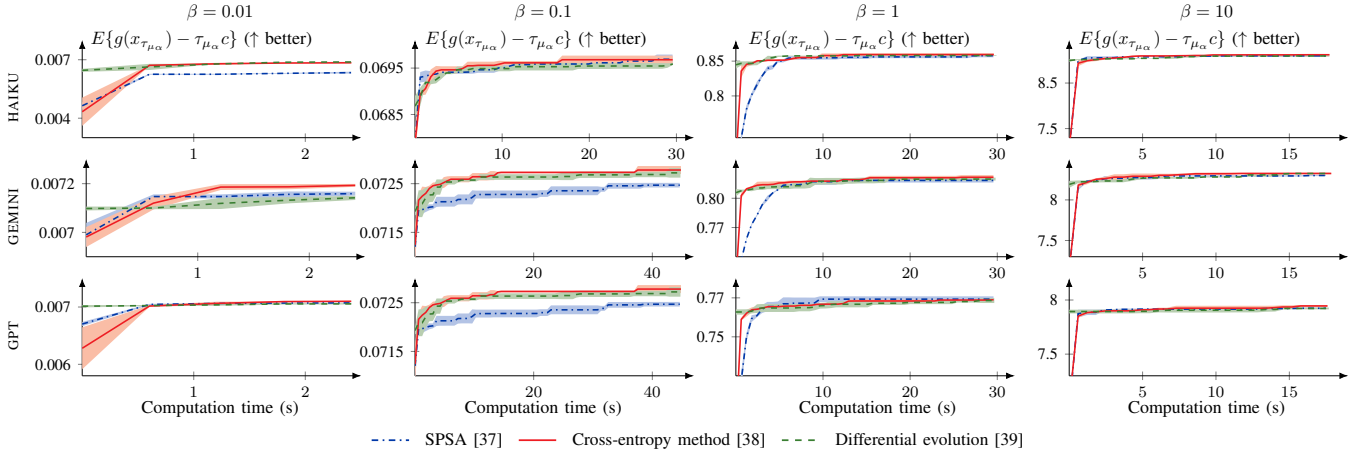


Fig. 5. Convergence curves of three simulation-based optimization methods (SPSA [37], Cross-entropy method [38], and Differential evolution [39]) for computing the optimal threshold α^* [cf. Prop. 4] across the three foundation models and four values of the payoff weight β (defined in §V-C). Curves show mean values and shaded regions indicate one standard deviation over 3 runs with different random seeds. The y-axes indicate the expected value $E\{g(x_{\tau_{\mu_\alpha}}) - c\tau_{\mu_\alpha}\}$ (i.e., the expected payoff minus the refinement costs) and the x-axes indicate the computation times in seconds.

Proof. By Prop. 3, we have $\mathcal{S}_k^* \subseteq \mathcal{S}_{k+1}^*$ for all stages $k \in \{0, 1, \dots, N-2\}$. It remains to show the reverse inclusion. Fix $k \in \{0, 1, \dots, N-2\}$ and let $x \in \mathcal{S}_{k+1}^*$, so that $x \geq \alpha_{k+1}$ and $g(x) \geq Q_{k+1}^*(x)$. Since $f(x, w) = \min\{1, \max\{x, q(x) + w\}\} \geq x \geq \alpha_{k+1} \geq \alpha_{k+2}$ (where $\alpha_N = 0$ since stopping is enforced at stage N), we have $f(x, w) \in \mathcal{S}_{k+2}^*$ for all realizations of w , and therefore $V_{k+2}^*(f(x, w)) = g(f(x, w))$. Similarly, $f(x, w) \in \mathcal{S}_{k+1}^*$, so $V_{k+1}^*(f(x, w)) = g(f(x, w))$. It follows that

$$\begin{aligned} Q_k^*(x) &= -c + E\{V_{k+1}^*(f(x, w))\} = -c + E\{g(f(x, w))\} \\ &= -c + E\{V_{k+2}^*(f(x, w))\} = Q_{k+1}^*(x). \end{aligned}$$

Since $g(x) \geq Q_{k+1}^*(x) = Q_k^*(x)$, we have $x \in \mathcal{S}_k^*$. Thus $\mathcal{S}_{k+1}^* \subseteq \mathcal{S}_k^*$, which together with $\mathcal{S}_k^* \subseteq \mathcal{S}_{k+1}^*$ gives $\mathcal{S}_k^* = \mathcal{S}_{k+1}^*$ for every $k \in \{0, 1, \dots, N-2\}$. $\square$

Propositions 2–4 establish that, given the identified dynamics in Fig. 4 and assuming that Observations 1–2 are exact properties of the system function, an optimal stopping policy is characterized by a single threshold $\alpha^* \in [0, 1]$. This structure implies a simple method for computing an optimal policy: parameterize the stopping policy μ by a threshold α [cf. (8)] and estimate the optimal threshold α^* by maximizing the expected value $E\{g(x_{\tau_{\mu_\alpha}}) - c\tau_{\mu_\alpha}\}$, where μ_α denotes the parameterized policy. This optimization can be efficiently performed using simulation-based stochastic approximation methods, as detailed in the following section.

VII. COMPUTING AN OPTIMAL STOPPING POLICY

We now exploit the structural properties established in the preceding section to compute an optimal stopping policy. By Props. 2–4, this computation reduces to optimizing the stopping threshold α , which we use to parameterize a threshold-based stopping policy μ_α ; cf. (8). We implement this optimization using three methods: simultaneous perturbation stochastic approximation (SPSA) [37], the cross-entropy method [38], and differential evolution [39]. For

each method, the value $E\{g(x_{\tau_{\mu_\alpha}}) - c\tau_{\mu_\alpha}\}$ is estimated via simulation of the Gaussian process model identified in §V-C. We run all simulations and optimizations on an M4 Pro Chip with macOS Sequoia 15.6.1 and Python 3.11.

Figure 5 shows the convergence curves for varying values of the payoff weight β , as defined in §V-C. All methods converge to similar expected values, which suggests that they identify optimal or near-optimal thresholds. In all cases, convergence is achieved within seconds, which demonstrates the computational efficiency afforded by the threshold structure.

Figure 6 shows the optimized thresholds for each model and payoff weight β . We observe that the thresholds increase with β , which is expected. As the relative importance of payoff grows, the policy becomes more selective and demands higher scores before stopping. For small β (e.g., $\beta = 0.01$), the thresholds are low, indicating that the cost of refinement dominates and early stopping is preferred. Across the three foundation models, HAIKU 4.5 admits lower thresholds than the other models for small β , which is consistent with its higher continuation cost; cf. Table I. For large β , however, HAIKU 4.5 exhibits higher thresholds than the other models.

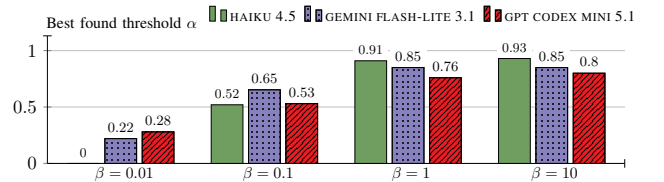


Fig. 6. The optimized threshold α across the three foundation models and four values of the payoff weight β (defined in §V-C).

VIII. EXPERIMENTAL EVALUATION ON EFFIBENCH

In this section, we evaluate the optimized stopping policy described in the preceding section on EFFIBENCH [33] and compare it against stopping policies proposed in prior work.

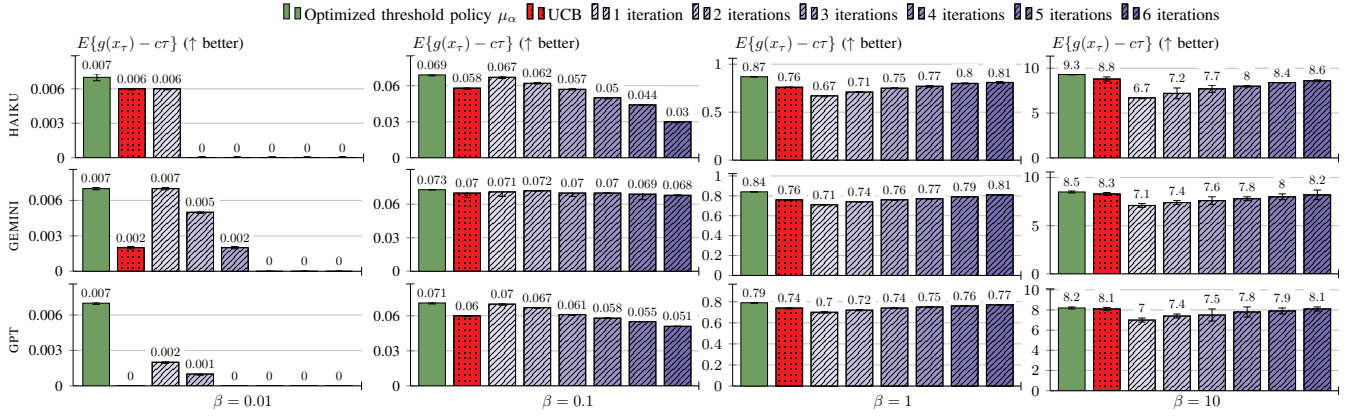


Fig. 7. Evaluation results on EFFIBENCH [33]. Each group of bars shows the expected value $E\{g(x_\tau) - c\tau\}$ for the optimized threshold policy μ_α , the UCB baseline, and fixed-iteration stopping policies with 1 to 6 iterations. Rows correspond to the three foundation models. Columns indicate the payoff weight β (defined in §V-C). Bar heights indicate the mean over evaluations with three different random seeds; error bars indicate one standard deviation.

Baseline stopping policies. We consider two categories of baselines. The first category consists of *fixed-iteration* policies, which stop after a predetermined number of iterations. This is the most common approach in the self-refinement literature; see e.g., [10], [9], [40], [41], [42]. The second category is the *upper confidence bound* (UCB) policy proposed by Sun et al. [43], which adaptively decides when to stop based on confidence bounds on the values of a finite set of stopping thresholds. We instantiate this policy by defining the set of thresholds as $\alpha \in \{0, 0.1, 0.2, \dots, 1\}$ and selecting the threshold with the highest upper confidence bound.

Experiment setup. We evaluate all stopping policies on 50 problems from EFFIBENCH [33]. (These problems are disjoint from the problems used for system identification in §V-C.) Each stopping policy is applied to the three foundation models with a maximum of $N = 10$ refinement iterations. We run each method three times with different random seeds and report the average performance.

Evaluation results. Figure 7 shows the expected value $E\{g(x_{\tau_\mu}) - c\tau_\mu\}$ for each stopping policy μ across the three foundation models and four values of the payoff weight β . We observe that the optimized threshold policy μ_α achieves the highest expected value in all configurations.

For the fixed-iteration policies, we find that the best policy varies with β . In particular, we observe that fewer iterations perform better when β is small, while more iterations are preferred when β is large. Finally, our experimental results show that the UCB policy automatically adapts to β but consistently underperforms the optimized threshold policy.

IX. DISCUSSION OF THE EXPERIMENTAL RESULTS

The main takeaways from the theoretical and experimental results can be summarized as follows.

- 1) *Self-refinement exhibits diminishing returns.* The empirical measurements in Fig. 3 show that all three foundation models improve output quality through self-refinement, but with progressively smaller gains per

refinement iteration. Meanwhile, token consumption and monetary cost grow approximately linearly.

- 2) *Optimal threshold-based stopping policies exist.* Propositions 2–4 establish that, under the conditions in Observations 1–2, an optimal stopping policy is characterized by a single, stage-independent threshold α^* . This reduces the policy search from a sequence of functions $\mu_0, \dots, \mu_{N-1}$ to a scalar optimization problem, which can be efficiently solved; see Fig. 5.
- 3) *Optimal stopping improves over the state-of-the-art.* The most common approach in the self-refinement literature is to stop after a fixed number of iterations. As shown in Fig. 7, the optimized threshold policy consistently outperforms this approach.

X. CONCLUSION

Foundation models can improve their outputs through a self-refinement process where they iteratively critique and refine their own outputs. We show that this process can be formulated as an optimal stopping problem in which the decision to continue refining or to stop is made sequentially based on the expected improvement relative to the refinement cost. We establish conditions for optimal threshold-based stopping policies and validate them on a coding benchmark across three frontier models. Empirical results demonstrate that our stopping policy significantly improves cost-efficiency compared to the state-of-the-art. These results show the value of leveraging decision theory for efficient control of foundation models. As these models are increasingly deployed in large-scale automation pipelines with thousands of daily invocations, we believe that our optimal stopping approach can translate into substantial monetary savings.

Future work. A natural direction for future work is to evaluate our approach on benchmark types other than coding benchmarks (e.g., mathematical reasoning benchmarks) and to instantiate it with foundation models other than large language models (e.g., time-series models or multi-modal models). From a theoretical perspective, our structural results (Props. 2–4) rely on Observations 1–2 holding exactly. Since

the system function is estimated via a Gaussian process, a promising direction for further analysis is to relax these assumptions by exploiting the posterior uncertainty of the GP to derive probabilistic guarantees on the threshold structure.

REFERENCES

- [1] J. He, C. Treude, and D. Lo, “LLM-based multi-agent systems for software engineering: Literature review, vision, and the road ahead,” *ACM Transactions on Software Engineering and Methodology*, vol. 34, no. 5, pp. 1–30, 2025.
- [2] C. Lu, C. Lu, R. T. Lange, Y. Yamada, S. Hu, J. Foerster, D. Ha, and J. Clune, “Towards end-to-end automation of AI research,” *Nature*, vol. 651, no. 8107, pp. 914–919, Mar 2026.
- [3] M. F. Maleki and R. Zhao, “Procedural content generation in games: A survey with insights on emerging LLM integration,” in *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, vol. 20, no. 1, 2024, pp. 167–178.
- [4] K. Hammar, T. Alpcan, and E. C. Lupu, “Incident response planning using a lightweight large language model with reduced hallucination,” in *33rd Annual Network and Distributed System Security Symposium, NDSS 2026, San Diego, California, USA, February 23-27, 2026*. The Internet Society, 2026.
- [5] Gemini Team, Google *et al.*, “Gemini: A family of highly capable multimodal models,” 2024.
- [6] S. P. Arango, P. Mercado, S. Kapoor, A. F. Ansari, L. Stella, H. Shen, H. H. J. Senetaire, A. C. Turkmen, O. Shchur, D. C. Maddix, M. Bohlke-Schneider, B. Wang, and S. S. Rangapuram, “ChronosX: Adapting pretrained time series models with exogenous variables,” in *Proceedings of The 28th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, Y. Li, S. Mandt, S. Agrawal, and E. Khan, Eds., vol. 258, PMLR, 03–05 May 2025, pp. 2242–2250.
- [7] J.-B. Alayrac *et al.*, “Flamingo: a visual language model for few-shot learning,” in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35. Curran Associates, Inc., 2022, pp. 23 716–23 736.
- [8] T. Brown *et al.*, “Language models are few-shot learners,” in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901.
- [9] A. Madaan, N. Tandon, P. Gupta, S. Hallinan, L. Gao, S. Wiegrefe, U. Alon, N. Dziri, S. Prabhume, Y. Yang, S. Gupta, B. P. Majumder, K. Hermann, S. Welleck, A. Yazdanbakhsh, and P. Clark, “Self-Refine: Iterative refinement with self-feedback,” in *Advances in Neural Information Processing Systems*, vol. 36, 2023, pp. 46 534–46 594.
- [10] D. Huang, J. Dai, H. Weng, P. Wu, Y. Qing, H. Cui, Z. Guo, and J. M. Zhang, “EffiLearner: Enhancing efficiency of generated code via self-optimization,” in *Advances in Neural Information Processing Systems*, A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, Eds., vol. 37. Curran Associates, Inc., 2024, pp. 84 482–84 522.
- [11] Y. Weng, M. Zhu, F. Xia, B. Li, S. He, S. Liu, B. Sun, K. Liu, and J. Zhao, “Large language models are better reasoners with self-verification,” in *Findings of the Association for Computational Linguistics: EMNLP 2023*, H. Bouamor, J. Pino, and K. Bali, Eds. Association for Computational Linguistics, Dec. 2023, pp. 2550–2575.
- [12] Y. Wang, Y. Wu, Z. Wei, S. Jegelka, and Y. Wang, “A theoretical understanding of self-correction through in-context alignment,” in *Advances in Neural Information Processing Systems*, vol. 37, 2024.
- [13] K. Hammar, “Supplementary material of the paper “Optimal Stopping of Self-Refining Foundation Models”,” 2026, code: https://github.com/Kim-Hammar/optimal_stopping_foundation_models.
- [14] A. Wald, *Sequential Analysis*. Wiley and Sons, New York, 1947.
- [15] A. N. Shirayev, *Optimal Stopping Rules*. Springer-Verlag Berlin, 2007, reprint of Russian edition from 1969.
- [16] Y. Chow, H. Robbins, and D. Siegmund, *Great expectations: The theory of optimal stopping*. Houghton Mifflin, 1971.
- [17] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, 3rd ed. Belmont, MA, USA: Athena Scientific, 2005, vol. I.
- [18] A. G. Tartakovsky, B. L. Rozovskii, R. B. Blažek, and H. Kim, “Detection of intrusions in information systems by sequential change-point methods,” *Statistical Methodology*, 2006.
- [19] K. Hammar and R. Stadler, “Intrusion prevention through optimal stopping,” *IEEE Transactions on Network and Service Management*, vol. 19, no. 3, pp. 2333–2348, 2022.
- [20] V. Krishnamurthy, *Partially Observed Markov Decision Processes: From Filtering to Controlled Sensing*. Cambridge University Press, 2016.
- [21] M. Rabi and K. H. Johansson, “Event-triggered strategies for industrial control over wireless networks,” in *Proceedings of the 4th Annual International Conference on Wireless Internet*, ser. WICON ’08, 2008.
- [22] V. V. Veeravalli, G. Fellouris, and G. V. Moustakides, “Quickest change detection with controlled sensing,” *IEEE Journal on Selected Areas in Information Theory*, vol. 5, pp. 1–11, 2024.
- [23] X. Ren, K. H. Johansson, and L. Shi, “Quickest change detection with observation scheduling,” *IEEE Transactions on Automatic Control*, vol. 62, no. 6, pp. 2635–2647, 2017.
- [24] G. Peng and Q. Zhu, “Sequential hypothesis testing game,” in *2020 54th Annual Conference on Information Sciences and Systems (CISS)*, 2020, pp. 1–6.
- [25] J. Milošević, D. Umsonst, H. Sandberg, and K. H. Johansson, “Quantifying the impact of cyber-attack strategies for control systems equipped with an anomaly detector,” in *2018 European Control Conference (ECC)*, 2018, pp. 331–337.
- [26] H. Liu, Y. Li, J. Mårtensson, L. Xie, and K. H. Johansson, “Reinforcement learning based approach for flip attack detection,” in *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE Press, 2020, p. 3212–3217.
- [27] K. Hammar, T. Li, R. Stadler, and Q. Zhu, “Adaptive security response strategies through conjectural online learning,” *IEEE Transactions on Information Forensics and Security*, vol. 20, pp. 4055–4070, 2025.
- [28] K. Hammar, Y. Li, T. Alpcan, E. C. Lupu, and D. P. Bertsekas, “Adaptive network security policies via belief aggregation and rollout,” *IEEE Transactions on Information Forensics and Security*, vol. 21, pp. 6544–6559, 2026.
- [29] K. Hammar and T. Alpcan, “Scalable solutions to zero-sum partially observable stochastic games through belief aggregation with approximation guarantees,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 40, no. 20, p. 17023–17031, Mar. 2026.
- [30] J. L. Snell, “Applications of martingale system theorems,” *Transactions of the American Mathematical Society*, vol. 73, 1952.
- [31] J. Bather, *Decision Theory: An Introduction to Dynamic Programming and Sequential Decisions*. USA: John Wiley and Sons, Inc., 2000.
- [32] E. B. Dynkin, “The optimum choice of the instant for stopping a Markov process,” *Sov. Math. Dokl.*, vol. 4, pp. 627–629, 1963, translation from Dokl. Akad. Nauk SSSR 150, 238–240 (1963).
- [33] D. Huang, Y. Qing, W. Shang, H. Cui, and J. M. Zhang, “EffiBench: Benchmarking the efficiency of automatically generated code,” in *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, vol. 37, 2024, pp. 11 506–11 544.
- [34] Anthropic, “The Claude 3 model family,” Tech. Rep., 2024.
- [35] OpenAI, J. Achiam, S. Adler *et al.*, “GPT-4 technical report,” 2024.
- [36] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, 2006.
- [37] J. C. Spall, “Multivariate stochastic approximation using a simultaneous perturbation gradient approximation,” *IEEE Transactions on Automatic Control*, vol. 37, no. 3, pp. 332–341, 1992.
- [38] R. Rubinstein, “The cross-entropy method for combinatorial and continuous optimization,” *Methodology and Computing in Applied Probability*, vol. 1, no. 2, pp. 127–190, 1999.
- [39] R. Storn and K. Price, “Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces,” *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, Dec 1997.
- [40] E. Zelikman, E. Lorch, L. Mackey, and A. T. Kalai, “Self-taught optimizer (STOP): Recursively self-improving code generation,” in *First Conference on Language Modeling (COLM)*, 2024.
- [41] Z. Zhou, J. Song, K. Yao, Z. Shu, and L. Ma, “ISR-LLM: Iterative self-refined large language model for long-horizon sequential task planning,” in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 2081–2088.
- [42] A. Piché, A. Milios, D. Bahdanau, and C. Pal, “LLMs can learn self-restraint through iterative self-reflection,” *Transactions on Machine Learning Research*, 2025.
- [43] R. Sun, W. Cheng, D. Li, H. Chen, and W. Wang, “Stop when enough: Adaptive early-stopping for chain-of-thought reasoning,” in *Proc. 64th Annual Meeting of the Association for Computational Linguistics (ACL)*, San Diego, CA, USA, 2026, pp. 27 250–27 268.