Optimal Schedules for Monitoring Anytime Algorithms

Lev Finkelstein and Shaul Markovitch
Computer Science Department
Technion, Haifa 32000
Israel
lev,shaulm@cs.technion.ac.il

Abstract

Monitoring anytime algorithms can significantly improve their performance. This work deals with the problem of off-line construction of monitoring schedules. We study a model where queries are submitted to the monitored process in order to detect satisfaction of a given goal predicate. The queries consume time from the monitored process, thus delaying the time of satisfying the goal condition. We present a formal model for this class of problems and provide a theoretical analysis of the class of optimal schedules. We then introduce an algorithm for constructing optimal monitoring schedules and prove its correctness. We continue with distribution-based analysis for common distributions, accompanied by experimental results. We also provide a theoretical comparison of our methodology with existing monitoring techniques.

1 Introduction

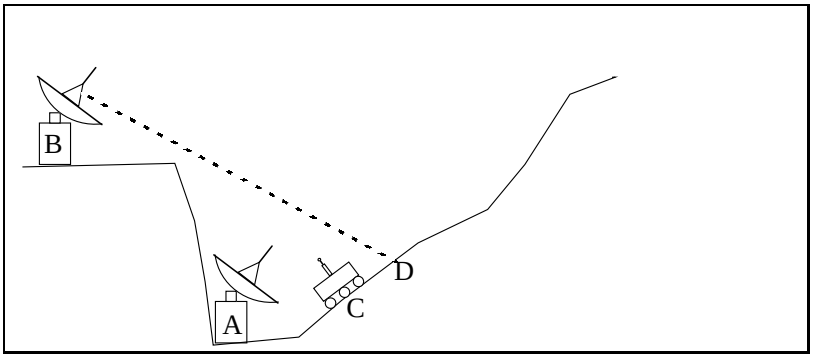


Figure 1: An example of the test scheduling problem: At which points should the robot stop in order to test communication?

Assume that two stations, A and B, attempt to communicate using laser-based transmission. The two stations do not have visual contact and thus cannot establish direct communication. A decides to send a receiver-transmitter robot C up the hill, as illustrated in Figure 1. The robot can initiate communication with the two stations starting from point D, which has visual contact with both stations. The robot must stop in order to establish communication.

If it stops at a point lower than D, it will not be able to communicate with B. However, since measurements of factors such as the robot's speed and position cannot be ascertained with precision, the time required for the robot to arrive at D can be evaluated only approximately. Therefore we preprogram the robot to stop at various points and test for communicability. Each test requires a constant time τ. Our goal is to generate a test schedule that minimizes the total time required to establish communication.

One possibility is to program the robot to stop after a time long enough to guarantee with high probability that the robot has passed point D. The problem with this approach is that on the average the robot will waste a lot of time traveling beyond D. An alternative approach is to program the robot to stop very often and perform its test. While this approach allows the robot to detect the reception area at an earlier time, the total time required to establish communication is still large due to the overhead of the tests. It seems that the correct approach lies somewhere between these two extremes. But is it possible to compute a test schedule that guarantees, on average, a minimal total time?

Another example of the test scheduling problem is a PROLOG interpreter. Assume that the interpreter processes a complex query and offers solutions to the user during the execution. The user visually examines each solution and responds either with a semicolon to continue the process or with a period to stop it. The time that the system spends waiting for the human response adds to the total execution time. Assume that we can estimate the number of solutions to the query and the time required to generate all of them[1]. Assume also that the interpreter is extended to allow presentation of more than one solution at a time and that there is a specific solution that the user is looking for. What policy provides the minimal expected total time for processing the query?

A third example is taken from the field of computational learning [16]. Assume that the goal of a concept learner is to PAC-learn a concept, i.e., with probability $1 - \delta$ to infer a hypothesis with a misclassification probability of less than ϵ. Assume that we know how to compute the minimal number of required examples based on ϵ and δ. Assume also that the learner is allowed to ask a weak form of *equivalence queries* [16], i.e., to ask the teacher whether our current hypothesis is correct[2]. Assuming that the cost of a query is constant, which policy would minimize the total learning time?

What do the above three examples have in common?

- An agent executes a process with the purpose of satisfying a given goal predicate.
- If the goal predicate is satisfied at time t^*, then it is also satisfied at any time $t > t^*$.
- The process can be queried at any time whether or not the goal predicate has been satisfied.
- During the query execution, the process is halted.
- The goal of the agent is to minimize the total time spent on the process, *including* the time spent for the queries, until the goal predicate is *known* to be satisfied.

[1] Ledeniov and Markovitch [17, 18] used similar information to increase the efficiency of a PROLOG interpreter by subgoal reordering. Such information can be learned by proving training queries. A learner can infer, for example, that the average number of solutions to a query of the class `parent(var,const)` is about 2.

[2] The regular equivalence queries require that a negative reply be accompanied by a counterexample.

The goal of this research is to develop algorithms that design an optimal query policy based on the statistical characteristics of the process. We begin by defining a formal framework for query-scheduling algorithms. The framework assumes a given statistical profile which describes the probability of the goal condition to be satisfied as a function of time. This profile is similar to *probabilistic performance profiles* [9], restricted to Boolean quality values. We then describe a sequence of intuitive query-scheduling algorithms and analyze their strengths and weaknesses. We continue with a general algorithm for an off-line calculation of an optimal query schedule and prove its optimality. We follow with distribution-based analysis that specializes the algorithm for uniform, exponential and normal distributions. This analysis is accompanied by solutions of the three example problems given above, including a formal analysis and an experimental evaluation using simulated data.

The idea of *monitoring* has received little attention within the AI research community, despite the fact that monitoring the state of an algorithm can significantly affect its performance. Monitoring is a subtopic of metalevel reasoning [22, 23] and has been studied primarily in the context of *anytime algorithms* [5, 10] and *contract algorithms* [24, 27]. The potential benefit of monitoring is to save computational resources of the monitored process. Monitoring itself, however, also carries a cost. This brings up the interesting question of when and how monitoring should be performed to optimize the tradeoff between its costs and benefits. Monitoring decisions can be therefore viewed as a kind of *type II rationality* [7], and the difference between the performance with and without monitoring corresponds to the concept of *intrinsic utility* [22].

Dean and Boddy [5, 2] have worked on a more complicated setup with a sequence of anytime algorithms. They assumed that no run-time monitoring is taking place and concentrated on the problem of finding a fixed resource allocation for each algorithm before it starts. They call this type of monitoring *deliberation scheduling*. The main input used in their works are *performance profiles* [25, 2] that measure the tradeoff between solution quality and computation time.

Horvitz [12] studied on-line monitoring extensively in the context of various application domains such as reformulation of belief networks [13, 3], automated theorem proving [14], and others. In the proposed models, the process stops when the expected benefit of halting is higher than the expected benefit of continuing computation. The domains described in these works have a higher degree of uncertainty than the model proposed here, allowing only myopic analysis of the tradeoffs involved. In the Protos system [11] Horvitz has extended the myopic horizon of EVC analysis by using a lookahead to a fixed depth. This scheme avoids some of the errors caused by myopic analysis.

Russell and Wefald [22, 21] describe a model of rational heuristic search. They propose an anytime algorithm for evaluating the expected utility of node expansion. The algorithm includes a stopping criterion enhanced by a monitoring procedure which tests the stopping criterion every fixed number of node expansions. This is an instance of the class of problems introduced above. A detailed analysis of this approach is given in Section 8.

The latest works of Zilberstein and Hansen [27, 8, 9] provide a theoretical framework for a wide range of monitoring problems, using a model with multiple-value quality and a high degree of uncertainty. Section 8 analyzes their work and compares it with our approach.

The rest of the paper is organized as follows. Section 2 describes intuitive approaches to the monitoring problem. Section 3 formulates the general framework used in this work. Sections 4 and 5 describe algorithms for generating optimal schedules. Section 6 contains distribution-specific analysis and offers solutions to the three problems above, along with

experimental evaluation on simulated data. Section 7 shows results for a problem using real data. Section 8 discusses related work and Section 9 presents our conclusions.

2 Intuitive approaches

For many problems like those described above, a human can produce a common-sense strategy. Assume that we are facing such a scheduling problem with a query cost of τ, and that we have an upper bound T on the time by which the goal is reached. In this section we present several intuitive strategies and show their weaknesses. The output of all the proposed methods will be a sequence of time points at which queries should be submitted.

There are two possible methods for representing a schedule. One is to specify the internal run time of the process (which does not include the query processing time). In that case the point of view expressed would be that of the process. The other method is to specify the total elapsed time, thus expressing the point of view of an external observer. A schedule represented by the first method as $\langle t_1, t_2, \ldots, t_n \rangle$ is equivalent to $\langle t_1, t_2 + \tau, \ldots, t_n + (n-1)\tau \rangle$ in the second method. In this paper we adopt the first method for representing schedules. Note, however, that regardless of the representation chosen, our goal here is to minimize the total elapsed time.

2.1 The query-at-the-end strategy

The simplest and therefore most common strategy is to query once when the allocated time T is exhausted. Such a strategy always requires a total time $T+\tau$. This approach is problematic

- **Input:** The maximum allowed time T.
- **Output:** $\langle T \rangle$.

Figure 2: The query-at-the-end algorithm.

when the expected time for satisfying the goal predicate is much less than T. For example, most of the classification learning algorithms accept a set of examples and process them all to get a classifier[3]. Since learning time is often greater than testing time, and since the required quality may be achieved with a much smaller set of examples, the query-at-the-end algorithm may produce sub-optimal behavior.

2.2 The query-every-Δt strategy

The problem with the former approach was the possible late detection of the goal condition. An alternative approach is to submit a query every Δt time units, where Δt can be as small as desirable. When $\Delta t = T$, we get the query-at-the-end strategy. The other extreme is to query after each atomic operation of the algorithm. Such a policy will solve the problem of late detection of the goal condition. However, if query cost τ is high, then a small Δt will be

[3] A notable exception is that of the windowing-based strategies such as those proposed by Quinlan [20] and by Fuernkranz [6]. There, a hypothesis is generated based on a portion of the examples. The learning is continued only if the classifier is not of the desired quality.

- **Input:** The maximum allowed time T, between-queries interval Δt.
- **Output:** $\langle \Delta t, 2\Delta t, \ldots, \left\lceil \frac{T}{\Delta t} \right\rceil \Delta t \rangle$.

Figure 3: The query-every-Δt algorithm

detrimental since the benefit of an early detection of the goal criterion will be outweighed by the added costs of the queries.

This approach is used, for example, by PROLOG interpreters, which ask for user confirmation after each solution is found. A less extreme approach is taken by Internet search engines, which usually return results to the user in chunks of 10 or 25.

2.3 The query-best-n-times strategies

Since querying at the end carries the danger of late detection and querying after each atomic operation carries the risk of high cumulative query cost, it seems reasonable to use the former strategy with an optimal number of queries. If we are given a distribution function $F(t)$ over the time when the goal predicate is satisfied, we can find the number of queries N that minimizes the expected total time. The algorithm implementing this approach, which we call QBN_t, is shown in Figure 4. Such a strategy, however, will be especially ineffective

- **Input:** Maximal allowed time T.
- **Algorithm:**
  1. Denote $\mathcal{T}_n = \langle \frac{T}{n}, \frac{2T}{n}, \ldots, \frac{(n-1)T}{n}, T \rangle$.
  2. Perform global minimization by n of the expected elapsed time of the process[a].
  3. Set N to be the optimal value of n.
  4. Return $\langle \frac{T}{N}, \frac{2T}{N}, \ldots, T \rangle$.

[a] More formally, we minimize $E(\mathcal{T}_n)$. E will be defined in Equation (3) in the following section.

Figure 4: The QBN_t strategy

when the probability of the goal predicate being satisfied is not uniformly distributed over $[0, T]$. In such cases a schedule with non-equal intervals can yield much better results than the optimal equal-step schedule. If, for example, this distribution is skewed towards T, then it is reasonable to query more often when approaching T.

This case can be handled by another strategy which equalizes the probability that the goal predicate will be satisfied between each two subsequent queries, i.e., $F(t_i) - F(t_{i-1}) = F(T)/n$. The strategy, which we call QBN_F, is described in Figure 5. One problem with the above approaches is their inability to handle tasks that are not limited in time. Another problem is that the schedules produced using these methods are not optimal. In the following sections

- **Input:** Maximal allowed time T.
- **Algorithm:**
  1. Denote $\mathcal{T}_n = \left\langle F^{-1}\left(\frac{F(T)}{n}\right), F^{-1}\left(\frac{2F(T)}{n}\right), \ldots, F^{-1}\left(\frac{(n-1)F(T)}{n}\right), T\right\rangle$.
  2. Perform global minimization by n of the expected elapsed time of the process.
  3. Set N to be the optimal value of n.
  4. Return $\left\langle F^{-1}\left(\frac{F(T)}{N}\right), F^{-1}\left(\frac{2F(T)}{N}\right), \ldots, T\right\rangle$

Figure 5: The QBN_F strategy

we propose a methodology for constructing optimal schedules which can also handle time-unlimited tasks.

3 A framework for off-line query scheduling

In this section we formalize the intuitive description of the query-scheduling problem given in the introduction. Let $\mathcal{S}$ be a set of states. Let t be a time variable with non-negative real values. Let $\mathcal{A}$ be a random process such that each realization (trajectory) $A(t)$ of $\mathcal{A}$ represents a mapping from $\mathcal{R}^+$ to $\mathcal{S}$.

Let $G : \mathcal{S} \rightarrow \{0, 1\}$ be a *goal predicate*, where 0 corresponds to $False$ and 1 corresponds to $True$. We say that $\mathcal{A}$ is *monotonic* over G if and only if for each trajectory $A(t)$ of $\mathcal{A}$ the function $\widehat{G_A}(t) = G(A(t))$ is a non-decreasing function. Under the above assumptions, $\widehat{G_A}(t)$ is a step function with at most one discontinuity point. $\widehat{G_A}(t)$ describes the behavior of the goal predicate as a function of time for a particular realization of the random process.

This scheme resembles the one used in anytime algorithms. The goal predicate can be viewed as a special case of the quality measurement used in anytime algorithms, and the requirement for its non-decreasing value is a standard requirement of these algorithms. The trajectories of $\mathcal{A}$ correspond to conditional performance profiles [28, 27]. However, the nature of the problem requires that we use a *cost* function $u(t)$ instead of the *utility* function commonly used in the anytime algorithms literature. We assume that u is a monotonic non-decreasing function.

Let $\mathcal{A}$ be monotonic over G. The definitions above show that the behavior of G for each trajectory $A(t)$ of $\mathcal{A}$ can be described by a single point $\widehat{t}_{A,G}$, the first point after which the goal predicate is true, i.e, $\widehat{t}_{A,G} = \inf_t\{t|\widehat{G_A}(t) = 1\}$. If $\widehat{G_A}(t)$ is always 0, we say that $\widehat{t}_{A,G}$ is not defined. Therefore, we can define a random variable $\zeta = \zeta_{\mathcal{A},G}$, which for each trajectory $A(t)$ of $\mathcal{A}$ with $\widehat{t}_{A,G}$ defined, corresponds to $\widehat{t}_{A,G}$.

The behavior of ζ can be described by its distribution function $F(t)$. At the points where $F(t)$ is differentiable, we use the probability density $f(t) = F'(t)$.

It is important to note that in practice not each trajectory of $\mathcal{A}$ leads to goal predicate satisfaction even after infinitely large time. That means that the set of trajectories where $\widehat{t}_{A,G}$ is undefined is not necessarily of measure zero. That is why we define the *probability of*

success p as the probability of $A(t)$ with $\widehat{t}_{A,G}$ defined [4].

For some problems, a time limit T on the running time of the process is given. We call such problems *time-limited*. Otherwise we call the problems *time-unlimited* and define T to be ∞.

Definition 1 *A* query *is a procedure that, when applied at time t, performs the following actions:*

1. *Suspends process A.*
2. *Computes the goal predicate at t.*
3. *If $\widehat{G_A}(t) = 0$ and $t < T$, the query resumes the algorithm. Otherwise it is stopped.*

The time during which the algorithm has been suspended is denoted by τ, and the cost of additional resources required for a single query application is denoted by C. We assume that C is expressed in the same units as $u(t)$. In the current model we assume both τ and C to be non-negative constants.

Definition 2 *We define a* schedule $\mathcal{T}$ *as a non-decreasing sequence of time points $\langle t_1, t_2, \ldots, t_n\rangle$ (or $\langle t_1, t_2, \ldots, t_k, \ldots\rangle$ for the infinite case).*

A schedule is used in the following way: At each time point t_i in the schedule $\mathcal{T}$ a query is applied to the process starting from t_1. If the goal predicate is satisfied or $t_i \geq T$, the process stops. Otherwise the process resumes. The whole procedure stops either when the process is stopped by the query or (in the case of finite schedules) t_n is passed.

Our framework assumes that satisfying the goal predicate is useful only if it is detected by a query. Therefore we require that at least one element of a schedule for the time-limited case will not be less than T. This implies $t_n \geq T$. In addition, from the definition of query 1 given above, $t_{n-1} < T$ (otherwise the process would always stop after t_{n-1}). The above observations lead to the following constraints over schedules for time-unlimited problems:

$$t_0 = 0 \leq t_1 \leq t_2 \leq \ldots \leq t_{n-1} < T \leq t_n < \infty. \tag{1}$$

Definition 3 *We define the* stopping time *of schedule $\mathcal{T}$ with respect to process realization A as the first point $t^* \in \mathcal{T}$, such that either $\widehat{G_A}(t^*) = 1$ or $t^* \geq T$. If no $t \in \mathcal{T}$ satisfies this condition, we say that $t^* = \infty$.*

From the above definition it follows that the cost $u_A(\mathcal{T})$ of schedule $\mathcal{T}$ for process realization A with a stopping point $t^* = t_k$ is

$$u_A(\mathcal{T}) = u(t_k + k\tau) + kC. \tag{2}$$

Note that u is not necessarily linear and therefore the above expression cannot be replaced by $u(t_k) + k(u(\tau) + C)$. Let $\mathcal{T} = \langle t_1, t_2, \ldots, t_n\rangle$ be a finite schedule[5]. Let us denote by t_0 the start time of the process, i.e., $t_0 = 0$. Let F be a distribution function over ζ and p be

[4] Another way to express the possibility that the process will not stop at all is to use profiles that approach $1 - p$ when $t \to \infty$. We prefer to use p explicitly because, in order for F to be a distribution function, it must satisfy $\lim_{t\to\infty} F(t) = 1$.

[5] The case of infinite schedules will be analyzed later.

the probability of success. The probability of the goal predicate being satisfied in the time segment from t_{i-1} to t_i is equal to $p(F(t_i) - F(t_{i-1}))$. The cost associated with this event is $u(t_i + i\tau) + iC$. The probability of the goal predicate being satisfied after t_n is $1 - pF(t_n)$, and the associated cost is $u(t_n + n\tau) + nC$. Therefore, the *expected* cost of schedule $\mathcal{T}$ with respect to F and p is

$$Eu(\mathcal{T}) = Eu(t_1, \ldots, t_n) = \\ p\left[\sum_{i=1}^{n} (u(t_i + i\tau) + iC)(F(t_i) - F(t_{i-1}))\right] + (1 - pF(t_n))(u(t_n + n\tau) + nC). \quad (3)$$

In the future we denote $Eu(\mathcal{T})$ by $E(\mathcal{T})$. Sometimes it will be more convenient to use an alternative formulation of (3) which can be obtained by a simple regrouping of terms:

$$Eu(t_1, \ldots, t_n) = u(t_n + n\tau) + nC - p\sum_{i=1}^{n-1} (u(t_{i+1} + (i+1)\tau) - u(t_i + i\tau) + C)F(t_i). \quad (4)$$

Our goal is to find a schedule with minimal expected cost. That means that we must choose a number n and a time schedule $\mathcal{T} = \langle t_1, \ldots, t_n \rangle$, such that $E(\mathcal{T})$ will be minimal. Thus, we must minimize (3) under the constraints given in (1).

Definition 4 *A schedule $\mathcal{T}_n$ is* optimal with respect to n *if it minimizes the value of $E(\mathcal{T})$ under the following constraints:*

$$t_0 = 0 \leq t_1 \leq t_2 \leq \ldots \leq t_{m-1} < T \leq t_m < \infty$$

with $m \leq n$. We call the corresponding value of E the optimal expected value for n, *and denote it by E^n_{opt}.*

The global optimal expected value, *E_{opt}, is defined as $\inf_n\{E^n_{opt}\}$. If there exists n such that the schedule $\mathcal{T}_n$ realizes E_{opt}, i.e. $E(\mathcal{T}_n) = E_{opt}$, we call $\mathcal{T}_n$* a global optimal schedule *and denote it by $\mathcal{T}_{opt}$.*

A schedule $\mathcal{T}$ is defined to be ϵ-optimal *if $E(\mathcal{T}) - E_{opt} < \epsilon$.*

If F is differentiable, we can rewrite (3) in another form:

$$Eu(t_1, \ldots, t_n) = p\sum_{i=1}^{n} \int_{t_{i-1}}^{t_i} (u(t_i + i\tau) + iC)f(t)dt + (1 - pF(t_n))(u(t_n + n\tau) + nC). \quad (5)$$

The form above is the specific case of the equation

$$Eu(t_1, \ldots, t_n) = p\sum_{i=1}^{n} \int_{t_{i-1}}^{t_i} u(t, t_i, i, \tau, C)f(t)dt + (1 - pF(t_n))u(t_n, t_n, n, \tau, C), \quad (6)$$

corresponding to the case when u can depend on t itself, for example, when the penalty is set for missing the exact moment when the goal predicate holds.

A lower limit on the expected schedule cost (which determines an upper limit on the possible savings) is obtained from (5) by setting $\tau = 0$ and $C = 0$.

$$E(t) \geq p\int_{t_0}^{t_n} u(t)f(t)dt + (1 - pF(t_n))u(t_n). \quad (7)$$

This case represents a pure off-line control, where queries use no resources.

In the following section we present an algorithm for finding an ϵ-optimal schedule for time-limited problems. Section 5 describes a similar algorithm for time-unlimited problems.

4 An optimal scheduling algorithm for time-limited problems

In this section we present an algorithm for finding an ϵ-optimal schedule. We start by proving necessary conditions for schedule optimality and continue with a theorem about sufficient conditions for the existence of a global optimal schedule. We then specify a method for finding the first element of a globally optimal schedule and a recursive formula to construct the rest of the sequence. We present an algorithm that combines the recursive formula with a standard single-variable optimization method and prove that this algorithm is guaranteed to find an ϵ-optimal schedule.

4.1 Properties of optimal schedules

In the analysis below we assume that F and u have first derivatives and u is a monotonic increasing function. In the Appendix we will show how to weaken these assumptions. In addition, we assume that either τ or C is not zero[6]. We also assume that the probability of success, p, is positive. If it is zero, then there is no sense in querying the process at all. Our last assumption is that $F(t)$ is strictly smaller than $F(T)$ for each $t < T$. Otherwise, there exists $t' < T$ such that F is constant over the segment $[t', T]$, and there is no sense in querying after t', which means that condition (1) is too strong.

4.1.1 Necessary conditions for schedule optimality

Before we proceed to our main theorem, we prove three properties of optimal schedules.

Lemma 1 *Let $\mathcal{T} = \langle t_1, \ldots, t_n \rangle$ be an optimal schedule. Then the following conditions hold:*

$$t_i \neq t_{i+1} \text{ for } i = 0, \ldots, n-1 \quad (8)$$

$$F(t_i) \neq F(t_{i+1}) \text{ for } i = 0, \ldots, n-1 \quad (9)$$

$$t_n = T. \quad (10)$$

Intuitively, the lemma means that if a goal predicate cannot be satisfied between two time points, then there is no need to query at both points.

Proof: We first want to show how eliminating a single point from a schedule affects the expected cost. Let $\mathcal{T}' = \langle t_1, t_2, \ldots, t_{i-1}, t_{i+1}, t_n \rangle$ be a schedule obtained from $\mathcal{T}$ by eliminating t_i. By (3) we can see that the difference between the expected costs of these schedules can be written as

$$\begin{aligned} &E(\mathcal{T}) - E(\mathcal{T}') = \\ &p \cdot [(u(t_i + i\tau) + iC)(F(t_i) - F(t_{i-1})) + (u(t_{i+1} + (i+1)\tau) + (i+1)C)(F(t_{i+1}) - F(t_i)) - \\ &(u(t_{i+1} + i\tau) + iC)(F(t_{i+1}) - F(t_{i-1})) + \\ &\sum_{j=i+2}^{n} (u(t_j + j\tau) - u(t_j + (j-1)\tau) + C)(F(t_j) - F(t_{j-1}))]. \end{aligned} \quad (11)$$

From the assumptions about $F(t)$ given in the beginning of this subsection and the condition $t_{n-1} < T \leq t_n$ of (1), it immediately follows that $F(t_{n-1}) < F(t_n)$. This proves (9) for the case of $i = n - 1$.

[6]Otherwise no global optimal schedule exists (since any given schedule can be improved by adding new queries).

Assume now that there exists $1 \leq i \leq n-1$ such that $F(t_{i-1}) = F(t_i)$ [7]. Let us choose the largest i satisfying this condition and let $\mathcal{T}'$ be $\mathcal{T}$ with t_i eliminated. Using the fact that $F(t_{i-1}) = F(t_i)$, we obtain from (11) that

$$\begin{aligned}
&E(\mathcal{T}) - E(\mathcal{T}') = \\
&p \cdot [(u(t_{i+1} + (i+1)\tau) - u(t_{i+1} + i\tau) + C)(F(t_{i+1}) - F(t_i)) - \\
&\sum_{j=i+2}^{n} (u(t_j + j\tau) - u(t_j + (j-1)\tau) + C)(F(t_j) - F(t_{j-1}))] = \\
&p \cdot \sum_{j=i+1}^{n} (u(t_j + j\tau) - u(t_j + (j-1)\tau) + C)(F(t_j) - F(t_{j-1}))]. \qquad (12)
\end{aligned}$$

We see that u is an increasing function, either C or τ is positive, and $F(t_n) > F(t_{n-1})$; Therefore, $E(\mathcal{T}) - E(\mathcal{T}') > 0$. In other words, eliminating t_i improves the schedule, which contradicts the optimality of $\mathcal{T}$. This ends the proof of (9). (8) follows immediately from (9).

Let us now show that $t_n = T$. Indeed, by (1) we know that $t_n \geq T$. By (3) we see that the part of $E(\mathcal{T})$ affected by t_n can be written as:

$$\begin{aligned}
&p(u(t_n + n\tau) + nC)(F(t_n) - F(t_{n-1})) + (1 - pF(t_n))(u(t_n + n\tau) + nC) = \\
&(u(t_n + n\tau) + nC)(1 - pF(t_{n-1})). \qquad (13)
\end{aligned}$$

Due to the fact that $u(t)$ is an increasing function, we immediately obtain that if $t_n > T$ then substituting T for t_n will decrease $E(\mathcal{T})$. This proves the last part of the lemma. □

Corollary 1 *The following equation follows immediately from (7) and (10).*

$$E(t) \geq p \int_{t_0}^{T} u(t) f(t) dt + (1 - pF(T))u(T). \qquad (14)$$

The following theorem provides a set of tight constraints over optimal schedules.

Theorem 1 *(***The main theorem***) Let $\mathcal{T} = \langle t_1, \ldots, t_n \rangle$ be an optimal schedule with respect to n. Then for each $i = 1, \ldots, n-1$ the following equation holds:*

$$\frac{u(t_{i+1} + (i+1)\tau) - u(t_i + i\tau) + C}{u'(t_i + i\tau)} = \frac{F(t_i) - F(t_{i-1})}{F'(t_i)}. \qquad (15)$$

Proof: Since $\mathcal{T}$ is optimal for n, it minimizes $Eu(t_1, \ldots, t_n)$ over the polyhedral defined by (1) with borders specified by the equations $t_{i-1} = t_i$. According to (10), the optimization is performed over $n-1$ variables $t_1, \ldots, t_{n-1}$. By (8) $t_{i-1} \neq t_i$. Therefore, based on the differentiability of $Eu(t_1, \ldots, t_n)$ [8], the following equations hold in the points of minimum:

$$\frac{dE}{dt_i} = 0 \text{ for } i = 1, \ldots, n-1. \qquad (16)$$

By the differentiation of (3), we obtain

$$\begin{aligned}
&pu'(t_i + i\tau)(F(t_i) - F(t_{i-1})) - p(u(t_{i+1} + (i+1)\tau) + (i+1)C)F'(t_i) + \\
&p(u(t_i + i\tau) + iC)F'(t_i) = 0. \qquad (17)
\end{aligned}$$

[7] In order to use (11) as is, we shift the value of i by 1.

[8] E is differentiable due to the differentiability of F and u.

Since p is positive, after reordering of terms we get

$$(u(t_{i+1} + (i+1)\tau) - u(t_i + i\tau) + C)F'(t_i) = (F(t_i) - F(t_{i-1}))u'(t_i + i\tau). \tag{18}$$

Since u is a monotonic increasing function, $u'(t_i + i\tau) \neq 0$. Using this fact together with (9) and (18), we obtain that for optimal schedules with fixed n

$$F'(t_i) \neq 0. \tag{19}$$

This allows us to rewrite (18) as (15). □

The above theorem implies a method for generating optimal schedules as follows.

Theorem 2 *Given a first point, t_1, of an optimal time schedule, the rest of the points can be reconstructed in a unique way using the formula*

$$t_{i+1} = u^{-1}\left(u(t_i + i\tau) + \frac{F(t_i) - F(t_{i-1})}{F'(t_i)}u'(t_i + i\tau) - C\right) - (i+1)\tau. \tag{20}$$

Proof: The proof of the theorem follows immediately from (15). The uniqueness of u^{-1} is implied by u being a monotonic increasing function[9]. □

Let $\mathcal{T}_{t_1}$ denote the sequence obtained by applying (20) to t_1. We denote the family of *all* such sequences by $\mathcal{W}$. Theorem 2 claims that any optimal sequence belongs to $\mathcal{W}$. It is possible to show that members of $\mathcal{W}$ are not necessarily monotonically increasing[10] and therefore may not be legal schedules as defined by (1). The following proposition allows us to easily identify non-schedules in $\mathcal{W}$.

Proposition 1 *A necessary and sufficient condition for a sequence from $\mathcal{W}$ to be increasing from t_1 to t_n is $t_{n-1} < t_n$.*

Proof: It is obvious that the above condition is necessary. We will now prove by contradiction that it is sufficient. Assume that $t_{n-1} < t_n$ but there exists $i \in 2, \ldots, n$ such that $t_{i-1} \geq t_i$. Then, by applying the Mean Value Theorem to (15), we conclude that there exist points ξ in $[t_i + i\tau, t_{i+1} + (i+1)\tau]$ and η in $[t_{i-1}, t_i]$ such that

$$\frac{u'(\xi)(t_{i+1} - t_i + \tau) + C}{u'(t_i + i\tau)} = \frac{F'(\eta)(t_i - t_{i-1})}{F'(t_i)},$$

and therefore

$$t_{i+1} - t_i = \frac{F'(\eta)}{F'(t_i)}\frac{u'(t_i + i\tau)}{u'(\xi)}(t_i - t_{i-1}) - \frac{C}{u'(\xi)} - \tau. \tag{21}$$

From (21) and the fact that $u'(t) > 0$, $F'(t) \geq 0$, $F'(t_i) \neq 0$, and either C or τ is positive, we obtain that if $t_{i-1} \geq t_i$, then $t_i > t_{i+1}$. Thus, by induction, the rest of the sequence will be *decreasing*, in contradiction of our assumption. □

Finally, we show three important features of optimal schedules. The following proposition states that optimality is preserved for any linear combination of $u(t)$.

[9] u^{-1} need not be defined over the whole range $[0, \infty)$. For optimal schedules, according to Theorem 1, u^{-1} will be always applied correctly.

[10] See Section 6.1 for an example.

Proposition 2 *Let $F(t)$ be a distribution function, $C = 0$, $u(t)$ a time cost function and $\tilde{u}(t)$ a linear combination of $u(t)$, i.e., $\tilde{u}(t) = cu(t) + u_0$. Then if $\mathcal{T} = \langle t_1, \ldots, t_n \rangle$ is an optimal schedule for $u(t)$ with expected cost $E(\mathcal{T})$, it is also optimal for $\tilde{u}(t)$ with expected cost $cE(\mathcal{T}) + u_0$.*

The proof follows immediately from equations (4) and (15).

A commonly used time-cost function is $u(t) = t$. The following propositions hold for this case.

Proposition 3 *If $u(t) = t$, then without loss of generality we can consider $C = 0$.*

Proof: When $u(t) = t$, equation (20) becomes

$$t_{i+1} = t_i + \frac{F(t_i) - F(t_{i-1})}{F'(t_i)} - (C + \tau). \tag{22}$$

Substituting τ with $\tau + C$ reduces the problem to the case of $C = 0$. □

The last proposition describes the features of shifted distribution:

Proposition 4 *Let $u(t) = t$, $F(t)$ be a distribution function, and $\tilde{F}(t)$ a shifted distribution function, i.e., $\tilde{F}(t) = F(t - t'_0)$. If $\mathcal{T} = \langle t_1, \ldots, t_n \rangle$ is an optimal schedule for $F(t)$ with expected cost $E(\mathcal{T})$, then the schedule $\mathcal{T}' = \langle t_1 + t'_0, \ldots, t_n + t'_0 \rangle$ is optimal for $\tilde{F}(t)$ with expected cost $E(\mathcal{T}) + t'_0$.*

The proof follows from the previous proposition and equations (4) and (22).

4.1.2 Sufficient conditions for schedule optimality

Definition 4 defines the global optimal expected value, E_{opt}, as the infimum of the optimal expected values with respect to fixed number of queries. Note, however, that E_{opt} may not be realized by a schedule. It is possible that each schedule can be improved by adding new queries. In such a case, there exists a sequence of schedules such that their costs form a decreasing sequence converging to E_{opt}. Note, however, that for every ϵ there exists at least one global ϵ-optimal schedule. The following theorems specify sufficient conditions for the existence of a global optimal schedule, i.e., a schedule that realizes E_{opt}.

Theorem 3 *The problem of minimization of $E(\mathcal{T})$ under the constraints given in (1) for a fixed n always has at least one solution.*

Proof: From the assumptions on F and u, $E(\mathcal{T})$ is a continuous function. Since the constraints given in (1) describe a convex area, then, according to Weierstrass's theorem, E must achieve its minimum and maximum values in this area. □

Theorem 4 *If $C > 0$ and $\tau = 0$, there exists a global optimal schedule.*

Proof: We will use a proof by contradiction. Assume that no global optimal schedule exists. Thus the sequence $\{E(\mathcal{T}_1), E(\mathcal{T}_2), \ldots, E(\mathcal{T}_n), \ldots, \}$, where $\mathcal{T}_i$ is an optimal schedule with respect to i, is non-increasing[11] and converges to E_{opt}.

[11] Recall that $\mathcal{T}_i$ can contain less than i queries.

By (3) we obtain that $p = 1$ and $F(t_n) = 1$; otherwise the expected costs $E(\mathcal{T}_{n_i})$ would have grown to infinity when $n_i \to \infty$. Together with the fact that $u(t) \geq 0$, we obtain from (3) that

$$E(\mathcal{T}_n) > \sum_{i=1}^{n} iC(F(t_i) - F(t_{i-1})) = C\left[nF(t_n) - \sum_{i=0}^{n-1} F(t_i)\right] = C\sum_{i=0}^{n-1}(F(t_n) - F(t_i)).$$

Since $E(\mathcal{T}_1) \geq E(\mathcal{T}_n)$ for all n, and $E(\mathcal{T}_1) = u(T) + C$ we obtain that

$$u(T) + C = E(\mathcal{T}_1) \geq E(\mathcal{T}_n) > C\sum_{i=0}^{n-1}(F(t_n) - F(t_i)).$$

For every $\epsilon > 0$ and for each $\mathcal{T}_n$, the number of queries t_i such that $F(t_n) - F(t_i) > \epsilon$ can be at most $N_\epsilon = \frac{u(T)+C}{C\epsilon}$. Due to the fact that $t_n = T$ for optimal schedules, for large n all the queries of $\mathcal{T}_n$, except perhaps for the first N_ϵ, are grouped in an arbitrarily small neighborhood of T. Therefore, by Taylor's theorem, $F(t_{i-1}) = F(t_i) + (t_{i-1} - t_i)F'(t_i) + o(t_{i-1} - t_i)$. Thus, for large i in schedules with a large enough number of queries, it holds that

$$\frac{F(t_i) - F(t_{i-1})}{F'(t_i)} = t_i - t_{i-1} + o(t_i - t_{i-1}). \tag{23}$$

On the other hand,

$$\frac{u(t_{i+1}) - u(t_i) + C}{u'(t_i)} = t_{i+1} - t_i + o(t_{i+1} - t_i) + \frac{C}{u'(t_i)},$$

and therefore from (15) we obtain

$$t_{i+1} - t_i + o(t_{i+1} - t_i)\frac{C}{u'(t_i)} = t_i - t_{i-1} + o(t_i - t_{i-1}),$$

and thus

$$(t_{i+1} - t_i) + (t_{i-1} - t_i) + o(t_{i+1} - t_i) + o(t_i - t_{i-1}) = -\frac{C}{u'(t_i)}.$$

Since

$$\frac{C}{u'(t_i)} \geq \frac{C}{\max_{t\in[0,T]} u'(t)} > 0, \tag{24}$$

the right part of the equation is a strictly negative constant. The left part, however, is of the order $O(\epsilon)$, i.e., can be made arbitrarily small. This contradiction proves the theorem. □

Theorem 5 *If the following conditions hold:*

1. $\tau > 0$.
2. *Either* $\lim_{t\to\infty} u(t) = \infty$ *or* $C > 0$.
3. $\exists N, \delta > 0 : x > N \Rightarrow \frac{u(x+\tau)-u(x)}{u'(x)} \geq \delta$.

then there exists a global optimal schedule.

Proof: The proof is by contradiction. We will use a scheme similar to that of the previous theorem and show that the majority of time points of schedules with a large number of queries are concentrated near T. If $C > 0$, then this part is similar to the previous proof. Otherwise, since $\lim_{t\to\infty} u(t) = \infty$, by (3) we have $p = 1$ and $F(t_n) = 1$, and therefore

$$E(\mathcal{T}_n) \geq \sum_{i=1}^{n} u(t_i + i\tau)(F(t_i) - F(t_{i-1})).$$

Since $\lim_{t\to\infty} u(t) = \infty$, for each $\epsilon > 0$ there exists a number N_ϵ such that for each $i \geq N_\epsilon$

$$u(t_i + i\tau) > \frac{u(T+\tau)}{\epsilon},$$

and therefore for n large enough,

$$E(\mathcal{T}_n) \geq \sum_{i=k}^{n} \frac{u(t_n+\tau)}{\epsilon}(F(t_i) - F(t_{i-1})) = u(t_n+\tau)\frac{F(t_n) - F(t_k)}{\epsilon}.$$

Using the fact that $E(\mathcal{T}_n) \leq E(\mathcal{T}_1) = u(T+\tau)$, and for optimal solutions $t_n = T$, we obtain

$$F(t_n) - F(t_k) < \epsilon,$$

which means that, as in the previous case, all the queries of $\mathcal{T}_n$, except perhaps first N_ϵ, are grouped in an arbitrarily small neighborhood of T.

As before,

$$\frac{F(t_i) - F(t_{i-1})}{F'(t_i)} = t_i - t_{i-1} + o(t_i - t_{i-1}). \tag{25}$$

The right side of equation (15) for large i has the form

$$\frac{u(t_{i+1} + (i+1)\tau) - u(t_i + i\tau) + C}{u'(t_i + i\tau)} \geq \frac{u(t_{i+1} + (i+1)\tau) - u(t_i + i\tau)}{u'(t_i + i\tau)} =$$
$$\frac{u(t_{i+1} + (i+1)\tau) - u(t_{i+1} + i\tau)}{u'(t_i + i\tau)} + \frac{u(t_{i+1} + i\tau) - u(t_i + i\tau)}{u'(t_i + i\tau)} =$$
$$\frac{u(t_{i+1} + (i+1)\tau) - u(t_{i+1} + i\tau)}{u'(t_i + i\tau)} + t_{i+1} - t_i + o(t_{i+1} - t_i) \geq$$
$$t_{i+1} - t_i + o(t_{i+1} - t_i) + \delta.$$

As before, we obtain

$$t_{i+1} - 2t_i + t_{i-1} + o(t_{i+1} - t_i) + o(t_i - t_{i-1}) + \delta = 0,$$

which, as in the previous proof, leads to contradiction. □

We would like to point out that the third condition is not as strong as it might seem. Essentially it states that the utility function should behave reasonably well. For clarity, this condition was stated for the whole range. As the proof shows, we could weaken the condition to the neighborhood of the points of the form $T + i\tau$ for large i. The condition holds for convex down functions, since $u(x+\tau) - u(x) = u'(\xi)\tau$ for some point ξ between x and $x+\tau$, and, for such functions, $u'(x)$ is an increasing function. Moreover, all the functions satisfying

$$\lim_{x\to\infty} \frac{u(x+\tau) - u(x)}{u'(x)} \geq \delta > 0 \tag{26}$$

satisfy the third condition as well. It is easy to see (for example, using L'Hôpital's rule) that functions such as $u(t) = t$ and $u(t) = ln(t)$, which are often used as time cost functions, satisfy the third condition. Finally, it is clear from the proof that the third condition could be replaced by the condition

$$C > 0 \text{ and } \lim_{t \to \infty} u'(t) < \infty.$$

4.2 An algorithm for computing optimal schedules

One way of building an algorithm for finding an optimal schedule is to write a procedure for computing an optimal schedule with respect to a fixed n and optimize the expected cost over n. Equations (15) and (10) form a system of n equations with n variables. Section 6.1 uses this method for the case of $u(t) = t$ for uniform distribution. If the equations are non-linear, however, this method becomes infeasible in most cases.

Another way to obtain a solution is to use the fact that t_1 determines the rest of the schedule (see Theorem 2) and minimize $E(t)$ over *two* variables, t_1 and n. This algorithm, however, requires minimization of a function of two dependent variables[12], one of which can get only integer values. Optimization under such conditions is unstable. To rectify this problem we transform the above method to minimization of one variable function.

We define a function G that, given the value of the first time point, t_1, returns the expected cost of the member in $\mathcal{W}$ starting with t_1. The function starts with t_1 and works iteratively. At each iteration i, if t_i does not satisfy one of the necessary conditions of optimal schedules ($F'(t_i) = 0$, or u^{-1} is not defined for its argument in (20), or $t_i \leq t_{i-1}$), we declare the time sequence to be non-optimal, assign $G(t_1) = \infty$ and stop. If $t_i \geq T$, we define $G(t_1) = E(t_0, t_1, \ldots, t_{i-1}, T)$ and stop. We say in this case that the schedule $\langle t_1, \ldots, t_{i-1}, T \rangle$ is *produced* by the initial time point t_1. Otherwise, we calculate the value for t_{i+1} using equation (20), increase i by 1, and repeat the process. This algorithm is shown in Figure 6.

```
function G(t_1)
t_0 ← 0, i ← 1.
repeat
    if t_i does not satisfy one of the necessary conditions of optimal schedules then
        return ∞
    else if t_i ≥ T then
        return E(t_0, t_1, ..., t_{i-1}, T)
    else
        Calculate the value for t_{i+1} using equation (20)
        i ← i + 1
end repeat
```

Figure 6: An algorithm for finding the value for $G(t_1)$.

We want to prove the following theorem:

[12] t_1 depends implicitly on n because of boundary conditions.

Theorem 6 *The problem of global minimization of $G(t_1)$ by t_1 is equivalent to the global minimization of $E(\mathcal{T})$. In other words, if t'_1 provides the minimal value for $G(t_1)$ with the corresponding time sequence $\mathcal{T}' = \langle t'_1, t'_2, \ldots, t'_{n'} \rangle$, and $\mathcal{T}'' = \langle t''_1, t''_2, \ldots, t''_{n''} \rangle$ is a sequence providing the minimal value for E, then*

$$G(t'_1) = E(\mathcal{T}') = E(\mathcal{T}'') = G(t''_1).$$

Proof: Since $\mathcal{T}''$ is the optimal sequence for E, by (10) $t''_{n''} = T$. $\mathcal{T}''$ must satisfy equation (20), and therefore $E(\mathcal{T}'') = G(t''_1)$. t'_1 provides the minimal value for G, thus

$$G(t'_1) \leq G(t''_1) = E(\mathcal{T}'').$$

On the other hand, G is constructed such that $G(t'_1) = E(\mathcal{T}')$ and $G(t'_1) < \infty$. Since $\mathcal{T}''$ is an optimal sequence for E, we have

$$E(\mathcal{T}'') \leq E(\mathcal{T}') = G(t'_1),$$

which proves the theorem. □

Figure 7 lists a general algorithm for calculating an optimal time schedule. $arg\min_t G(t)$ is computed using one of the standard optimization methods (see for example [19]). By Theorem 6, this algorithm is guaranteed to find a global ϵ-optimal schedule. This is not necessarily the *exact* global minimum – even if one exists – due to computation errors in the minimization process.

$t_0 \leftarrow 0$.
$t_1 \leftarrow arg\min_t G(t)$.
$i \leftarrow 1$.
While $t_i < T$ **do begin**
  Calculate t_{i+1} from t_i and t_{i-1} using Formula (20).
  $i \leftarrow i + 1$.
end
$n \leftarrow i$.

$t_n \leftarrow T$ *Although for optimal schedules t_n is always equal to T, a computation error may give a slightly different result.*

Return the schedule $\mathcal{T} = \langle t_1, \ldots, t_n \rangle$.

Figure 7: An algorithm for finding an optimal schedule

The same algorithm can also be used for finding an optimal schedule with respect to a given n. To do so, we need to add to the calculation of $G(t_1)$ an additional stopping condition, $i > n$.

5 A query-scheduling algorithm for time-unlimited problems

The above scheme can be extended to handle cases with no time limit, i.e., $T = \infty$. If there exists a point T' such that $F(T') = 1$, the algorithm has probability 1 to stop before this

point. This reduces the problem to the time-limited case with $T = T'$. Therefore, we can assume that $F(t) < 1$. In such a case, a schedule cannot be finite since a finite schedule always has a positive probability of submitting the last query before the goal predicate is satisfied. By (3), the infinite schedule has a finite expected cost only when either $p = 1$, or both $u(\infty) < \infty$ and $C = 0$. Substituting these conditions into (3) we obtain that the expected cost in the first case will be

$$Eu(\mathcal{T}) = \sum_{i=1}^{\infty} (u(t_i + i\tau) + iC)(F(t_i) - F(t_{i-1})), \tag{27}$$

and

$$Eu(\mathcal{T}) = p \sum_{i=1}^{\infty} u(t_i + i\tau)(F(t_i) - F(t_{i-1})) + (1-p)u(\infty) \tag{28}$$

in the second case. In both cases the series must converge.

The optimality of schedules are equivalent to their finite analogs. The following theorem is the generalization of Theorem 1 to time-unlimited problems.

Theorem 7 *Let $\mathcal{T} = \langle t_1, t_2, \ldots, t_n, \ldots \rangle$ be an optimal schedule. Then for each $i \geq 1$ the following equation holds:*

$$\frac{u(t_{i+1} + (i+1)\tau) - u(t_i + i\tau) + C}{u'(t_i + i\tau)} = \frac{F(t_i) - F(t_{i-1})}{F'(t_i)}. \tag{29}$$

Proof: The proof is by contradiction. Suppose that $\mathcal{T}$ is optimal but for some $k \geq 1$ equation (29) does not hold. Let us look at the time-limited problem with $T = t_{k+1}$. By Theorem 1 the sequence $\langle t_1, t_2, \ldots, t_{k+1} \rangle$ cannot be optimal since it violates equation (15). Let $\langle t'_1, t'_2, \ldots, t'_{l'} \rangle$ be the optimal schedule for the time-limited problem with respect to $k+1$. By Lemma 1, $t'_{l'} = T = t_{k+1}$.

Therefore, for the time-limited problem

$$p \sum_{i=1}^{l'} (u(t'_i + i\tau) + iC)(F(t'_i) - F(t'_{i-1})) < p \sum_{i=1}^{k+1} (u(t_i + i\tau) + iC)(F(t_i) - F(t_{i-1})),$$

and from equations (27) and (28), it follows that changing the sub-sequence $\langle t_1, t_2, \ldots, t_{k+1} \rangle$ to $\langle t'_1, t'_2, \ldots, t'_{l'} \rangle$ would lower the expected cost. If so, then $\mathcal{T}$ is not the optimal schedule, which proves the theorem. □

As in the time-limited problem, the following theorem follows from Theorem 7:

Theorem 8 *Equation (20) hold for optimal sequences in the infinite case.*

It is easy to see that Theorem 2 and Propositions 2, 3 and 4 for the finite case are also correct for the infinite case.

To adapt the algorithm of the previous section to the time-unlimited case, we first define function $\tilde{G}(t_1)$, analogous to $G(t_1)$. However, since the expected cost is represented by a series, the algorithm implementing $\tilde{G}(t_1)$ must be provided with a convergence and divergence criterion. Both criteria get a prefix of the series. An example of a convergence criterion is a test of whether the last two elements differ by at most ϵ. An example of a divergence criterion is a test of whether the last element is greater than a given large number.

We define function $\tilde{G}(t_1)$ iteratively in the following way:

1. t_0 is set to 0 and i is set to 1.
2. The initial value of i is set to 1.
3. If t_i does not satisfy one of the necessary conditions of optimal schedules ($F'(t_i) = 0$, or u^{-1} is not defined on its argument in (29), or $t_i \leq t_{i-1}$), we declare the time sequence to be non-optimal, assign $\tilde{G}(t_1) = \infty$ and stop the calculation.
4. If the divergence criterion holds, we set $\tilde{G}(t_1) = \infty$ and stop the calculation.
5. If the convergence criterion holds, we set $\tilde{G}(t_1) = E(t_0, t_1, \ldots, t_i)$, and stop the calculation.
6. Otherwise we calculate the value for t_{i+1} using formula (29), increase i by 1, and return to step 3.

Theorem 6 holds for the time-unlimited case up to the correctness of the convergence and divergence criteria. The algorithm for the time-unlimited case, therefore, is similar to the algorithm for the time-limited case. Obviously, we cannot implement an algorithm that returns infinite schedules. Instead we assume that time points are returned one-by-one by request. The algorithm is shown in Figure 8. As for time-limited case, the quality of the solution depends on the minimization method, but theoretically the global ϵ-optimal solution will be found with any given ϵ.

$t_0 \leftarrow 0$.
$t_1 \leftarrow arg\min_t G(t)$.
$i \leftarrow 1$.
While (a new time point is required) **do begin**
    Calculate t_{i+1} from t_i and t_{i-1} using formula (29).
    $i \leftarrow i + 1$.
    Return t_i as the current query point[a].
end

[a] Unlike the time-limited problem, the time points are returned one by one.

Figure 8: An algorithm for finding an optimal strategy for time-unlimited problems

6 Distribution-based analysis

In Sections 4 and 5 we presented an analytical approach to the problem of optimal scheduling. Our approach reduces the optimization problem in the space of schedules to one-variable optimization which can be solved using standard numerical methods. In this section we perform a deeper analysis for standard distributions and show experimental results. For the analysis presented in this section, we assume the most common case where $u(t) = t$, i.e., the pure time minimization problem. Due to Propositions 3 and 4, we consider $t_0 = 0$ and $C = 0$.

6.1 Uniform distribution

In this subsection we present a full analytic solution for the uniform distribution model. We also show an application of the solution to the PROLOG example described in the introduction.

6.1.1 Formal solution

Assume that ζ (the random variable representing the time when the goal predicate becomes true) is uniformly distributed over the interval $[0, T]$, i.e., its distribution function, F, is

$$F(t) = \begin{cases} 0 & \text{if } t < 0 \\ t/T & \text{if } t \in [0, T] \\ 1 & \text{if } t > T \end{cases} \tag{30}$$

and its density function, f, is

$$f(t) = \begin{cases} 0 & \text{if } t \notin [0, T] \\ 1/T & \text{if } t \in [0, T]. \end{cases} \tag{31}$$

The following theorems specify the optimal schedule for the case of uniform distribution. The proofs are given in the Appendix.

Theorem 9 *Let $\mathcal{T} = \langle t_1, \ldots, t_n \rangle$ be a member of $\mathcal{W}$. Then:*

1. *The time points of $\mathcal{T}$ satisfy*
$$t_i = \frac{i}{n} T + \frac{i(n-i)}{2} \tau. \tag{32}$$

2. *A necessary and sufficient condition for this sequence to be non-decreasing is*
$$n \leq n_{max} = \left\lfloor \frac{1 + \sqrt{1 + \frac{8T}{\tau}}}{2} \right\rfloor. \tag{33}$$

3. *The expected cost of $\mathcal{T}$ is*
$$E(t_1, \ldots, t_n) = \left(1 - \frac{p}{2} + \frac{p}{2n}\right)(T + n\tau) - p \frac{n^3 - n}{24} \frac{\tau^2}{T}. \tag{34}$$

The easiest way to optimize the right side of (34) is by assuming the domain of n to be continuous.

Theorem 10 *The optimal value for the right side of (34) for continuous n is*

$$\xi_{opt} = \sqrt{\frac{1}{6} + \frac{2T}{\tau}\left(\frac{2}{p} - 1\right) - \sqrt{\frac{16T^2}{\tau^2}\left(\frac{1}{p} - 1\right)\frac{1}{p} + \frac{2}{3}\frac{T}{\tau}\left(\frac{2}{p} - 1\right) + \frac{1}{36}}}. \tag{35}$$

In the special case of $p = 1$

$$\xi_{opt} = \sqrt{\frac{1}{6} + \frac{2T}{\tau} - \sqrt{\frac{2}{3}\frac{T}{\tau} + \frac{1}{36}}}. \tag{36}$$

The optimal number of queries can be approximated by comparing the value of $E(\mathcal{T})$ for the sequences obtained by equation (32), with n having one of two values:

$$n_{opt} = max(\lfloor \xi_{opt} \rfloor, 1) \text{ or } n_{opt} = min(\lceil \xi_{opt} \rceil, n_{max}). \tag{37}$$

If we look for the optimal schedule with respect to a given N, we take the minimum between N and n_{opt}. This is based on the fact that the right part of (34) must achieve its minimum either in the points with zero derivative or on the boundaries.

6.1.2 The PROLOG example

In the introduction we presented an example of a monitoring problem in the context of PROLOG:

1. We assume that a query q has an associated set of solutions (bindings) denoted by $sol(q)$. We assume that the user is interested in exactly one solution q^* which can be recognized when observed.

2. The probability of q^* to be a member of $sol(q)$ is denoted by p.

3. We assume that the interpreter presents $sol(q)$ in chunks of possibly variable length. The user observes the proposed set of solutions and quits the process if the desired solution is found.

4. We assume that the cost of producing a chunk of length m is $c_1 m$ and the cost of its testing by the user is $c_2 m + \tau$, where c_2 is the the cost per solution and τ is the overhead per chunk.

5. We assume that we can estimate, based on past experience, the expected number of solutions, M_q, of a query q.

6. We assume that $sol(q)$ is arbitrarily ordered.

Our goal is to endow the interpreter with a decision algorithm for determining the sizes of chunks that should be presented to the user in order to minimize the total time of the process. We will now show how the general framework presented in the previous sections can be used to reach this goal.

By item 6 we conclude that this case is an instance of the uniform distribution model. Without loss of generality we can substitute c_1 with $c_1 + c_2$ and c_2 with zero, thus making this problem an instance of the general case with a fixed τ. It is easy to see that $T = M_q(c_1 + c_2)$. The algorithm returns an optimal schedule in terms of time points. Division by $c_1 + c_2$ allows us to easily translate them to chunk lengths. Since some of the problem's parameters are discrete whereas the model assumes continues data, we will assume during the solution that our parameters are continuous and will round the results at the end.

6.1.3 Simulation results

To illustrate the above analysis, we assigned some reasonable values to the parameters and computed the optimal schedules using the strategies discussed in previous sections. The expected size of $sol(q)$ is set to 20. p is set to 0.8, c_1 is set to 1 second, c_2 to 0.1 seconds and τ to 2 seconds.

The computed results are as follows:

- If a human views the results one by one (the regular PROLOG model), the expected total time will be 38.44 sec.
- If a human views all the results together, the time will be 24 sec.
- Using the optimal schedule, the optimal number of queries will be 3 (at the time points 8, 15 and 20), and the expected time will be 20.396 sec. To test for flaws in our computation, we also ran a simulation in which the location of q^* in the sequence was randomly generated. The average cost over 1000000 runs was 20.4021 seconds, which confirms the correctness of the analysis.

The advantage of the optimal method over the other two is obvious: when τ is very small, n_{opt} becomes very large, and the expected cost of a schedule generated by the optimal strategy will be close to $\left(1-\frac{p}{2}\right)T$, whereas checking after the last result has a constant value of $T+\tau$. When T is large and $\frac{T}{\tau}$ is small, only one check will be allowed, and the average cost of the optimal method will be $T+\tau$ instead of about $\left(1-\frac{p}{2}\right)(T+n\tau)$ in the case of checking after each result.

In another experiment, we tested the effect of the independent variables τ and p on the expected cost with fixed $T=100$. The four graphs in Figure 9 show the results obtained for the optimal, QBN_t and query-at-the-end methods. Each graph stands for a fixed value of p.

As can be seen from the graphs, the optimal strategy has much better performance than the query-at-the-end strategy. The advantage of the optimal strategy diminishes for small p since all the queries submitted before T are wasted whenever the process fails. The advantage grows for smaller τ since a small cost for a query enables a more condensed schedule, which detects the satisfaction of the goal condition earlier. The optimal strategy has only a small advantage over the QBN_t strategy.

6.2 Exponential distribution

We start with a discussion of some formal properties specific to the case of exponential distribution and continue with a solution of the computational learning example given in the introduction for both time-limited and time-unlimited problems.

6.2.1 Formal analysis

The exponential distribution is described by the density function

$$f(t)=\begin{cases} 0 & \text{if } t\le 0 \\ \lambda e^{-\lambda t} & \text{if } t>0, \end{cases} \tag{38}$$

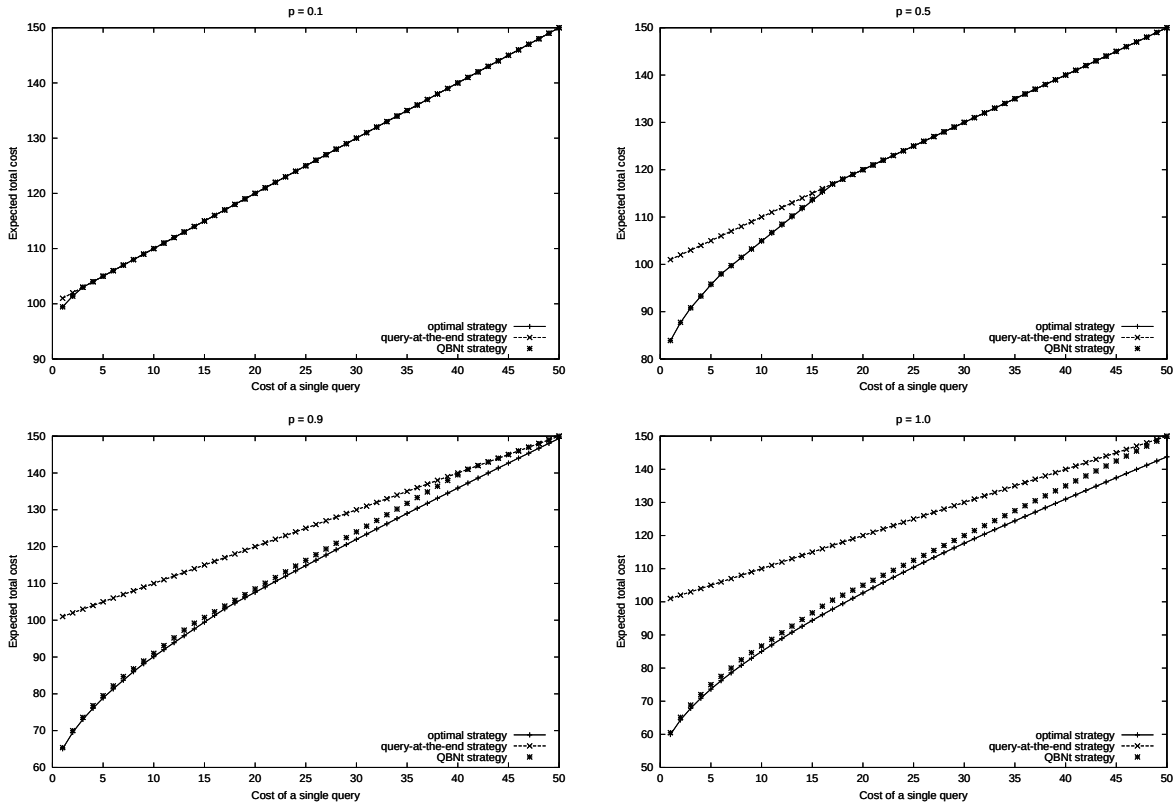


Figure 9: Each of the above graphs shows the performance of various scheduling methods as a function of the cost of a single query. The time T has been set to 100. The four graphs show the results for four different values of the probability of success: 0.1, 0.5, 0.9 and 1.0.

and since $F(0) = 0$, its distribution function has the form

$$F(t) = \begin{cases} 0 & \text{if } t \leq 0 \\ 1 - e^{-\lambda t} & \text{if } t > 0. \end{cases} \tag{39}$$

The theorem below describes the behavior of optimal schedules for exponential distributions:

Theorem 11 *Let us denote $g(x) = e^x - 1 - \lambda\tau$, $g_0(x) = x$ and $g_k(x) = g(g_{k-1}(x))$. Then for each fixed n, an optimal schedule $\mathcal{T} = \langle t_0 = 0, t_1, \ldots, t_n \rangle$ is described by a formula*

$$t_{i+1} = t_i + \frac{e^{\lambda(t_i - t_{i-1})} - 1}{\lambda} - \tau \text{ for } i = 1, \ldots, n-1, \tag{40}$$

where t_1 is the single root of the equation

$$t_1 + \frac{1}{\lambda} \sum_{i=1}^{n-1} g_i(\lambda t_1) = T, \tag{41}$$

and the corresponding expected cost of the process is

$$E(\mathcal{T}) = p\left((t_1 + \tau) + \frac{1}{\lambda}\left(1 - e^{-\lambda T + g_{n-1}(\lambda t_1)}\right)\right) + (1-p)(T + n\tau). \tag{42}$$

The proof of the theorem is given in the Appendix. We can utilize the above theorem as the basis for an alternative method for finding an optimal schedule for exponential distribution by minimization of $E(\mathcal{T})$ by n.

Corollary 2 *Let $\tilde{t_1}$ be a root of the equation*

$$\lambda t_1 = e^{\lambda t_1} - 1 - \lambda\tau. \tag{43}$$

Then the sequence of intervals between queries is increasing when $t_1 > \tilde{t_1}$, decreasing when $t_1 < \tilde{t_1}$ and constant when $t_1 = \tilde{t_1}$.

Proof: By (40) we obtain that the lengths of the time intervals satisfy the condition

$$\Delta_{i+1} = \frac{e^{\lambda\Delta_i} - 1}{\lambda} - \tau.$$

Assume that $\Delta_2 < \Delta_1$. Then

$$\Delta_3 = \frac{e^{\lambda\Delta_2} - 1}{\lambda} - \tau < \frac{e^{\lambda\Delta_1} - 1}{\lambda} - \tau = \Delta_2.$$

By induction, Δ_i will produce a decreasing sequence. The proof in the cases $\Delta_2 = \Delta_1$ and $\Delta_2 > \Delta_1$ is similar. □

6.2.2 The computational learning example

In this section we apply our algorithm to the computational learning problem described in the introduction. The monitored process is a learning-by-examples PAC-learning algorithm. We assume the framework of learning by a weak form of equivalence queries where the teacher only acknowledges or declines the correctness of the current hypothesis. The learning algorithm stops as soon as it get a positive reply to a query.

Assume that learning each example costs one unit of time and the cost of an equivalence query is τ units of time. The goal of the monitoring process is to design a query schedule for minimizing the overall time spent for learning the goal concept.

To apply our framework to this problem, we need the distribution function F, τ and T. We assume that $p = 1$ and that $u(t) = t$. The computational learning literature gives us an upper limit on the number of examples required for PAC-learning [26, 1]; this upper limit is based on ϵ, δ and the VC dimension of the concept class. Such dependencies can be used to infer both the distribution function F describing the behavior of the goal predicate and the time limit T.

Assume that our concept class is the set of axis-aligned rectangles over the Euclidean plane $\mathcal{R}^2$ and the examples are points drawn from $\mathcal{R}^2$. In [16] the authors show that after learning

$$m = \frac{4}{\epsilon} ln \frac{4}{\delta} \tag{44}$$

examples the probability that the model is ϵ-correct will be at least $1 - \delta$. We can therefore formulate the probability of being ϵ-correct as a function of m and ϵ:

$$1 - \delta(m) \geq 1 - 4e^{-\frac{m\epsilon}{4}}. \tag{45}$$

Since δ is a probability, we have $0 \leq 1 - \delta \leq 1$. The right part of the inequality (45) is less than or equal to 1 for positive m, but it is positive only for

$$m \geq m_0 = \frac{4ln4}{\epsilon}.$$

Therefore we can rewrite equation (45) as

$$1 - \delta(m) \geq 1 - e^{-\frac{\epsilon}{4}(m-m_0)}. \tag{46}$$

We will make the following assumptions:

- Since we have no better estimation for m, we will assume that m_0 is a tight lower bound, i. e., learning a smaller number of examples is assumed to be insufficient. Therefore, $1 - \delta(m) = 0$ for any $m \leq m_0$.
- We suppose that m has a continuous range, and we will denote m by t and m_0 by t_0. Since the total number of learned examples is usually large enough, the assumption about continuity will have no significant effect on the solution.

Now we can define the distribution function needed for our framework as:

$$F(t) = 1 - \delta(t) = \begin{cases} 0 & \text{if } t \leq t_0 \\ 1 - e^{-\frac{\epsilon}{4}(t-t_0)} & \text{if } t > 0, \end{cases} \tag{47}$$

where

$$t_0 = \frac{4ln4}{\epsilon}.$$

For given ϵ and δ we can easily compute T, the maximal number of required examples:

$$T = m_\epsilon(\delta) = \frac{4}{\epsilon} ln \frac{4}{\delta}.$$

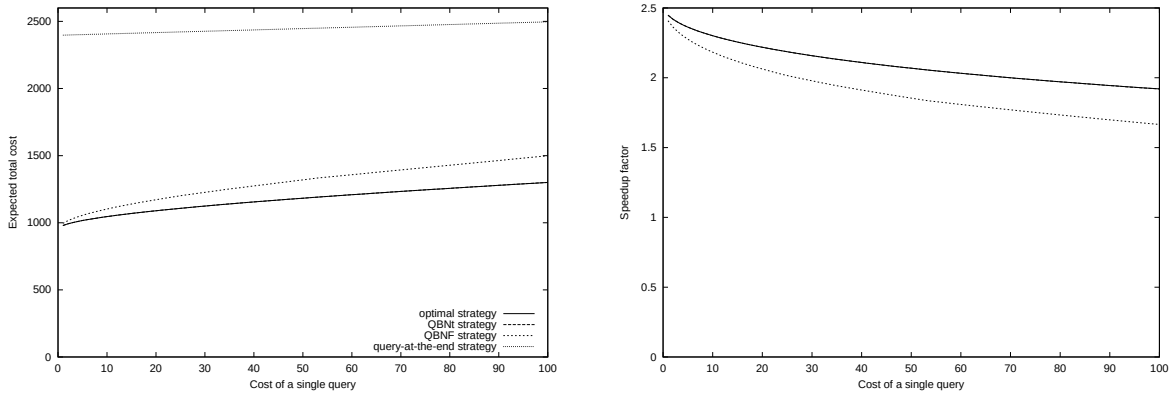
This problem is a specific case of the exponential distribution and can be solved either by the methods described in Section 4.2 or those described in Section 6.2. Our framework also allows us to design a monitoring strategy for the case of $\delta \to 0$, which stands for the time-unlimited case as described in Section 5.

6.2.3 Simulation results

We ran a set of experiments to test the effect of the independent variables τ, ϵ and δ on the expected total cost $E(\mathcal{T})$. The test was run with four different scheduling strategies: the optimal algorithm, the QBN_t strategy, the QBN_F strategy and the query-at-the-end strategy. In addition to the absolute expected total cost, we also show the speedup factor of the optimal method over the query-at-the-end method. τ was varied between 1 and 100 with a default value of 10. ϵ and δ were each varied between 0.01 and 1 with a default value of 0.01.

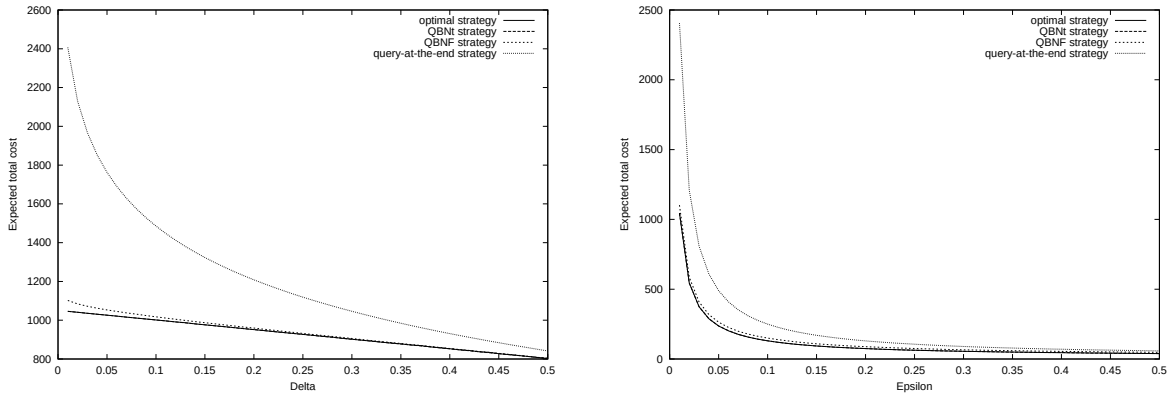
The results[13] are shown in Figures 10,11 and 12.

[13] In the experiments here and below we used the software for Brent optimization [4] written by Oleg Keselyov, available in Netlib public access repository at http://www.netlib.org.



(a) (b)

Figure 10: (a) The expected total cost as a function of the cost of a single query for various scheduling strategies. (b) The speedup factor of the three more sophisticated methods relative to the query-at-the-end method as a function of τ.



(a) (b)

Figure 11: (a) The expected total cost as a function of ϵ. (b)The expected total cost as a function of δ.

Figure 10 describes the total expected cost as a function of τ. We can see that for small τ the optimal method achieves a speedup factor of about 2.5 over the query-at-the-end method. It is interesting to note that the QBN_t method produces results which are almost equivalent to those achieved by the optimal method. This is a characteristic of left-skewed distributions where the overhead of the extra queries at the tail is offset by the low probability of their occurrence.

Figure 11 shows the cost as a function of δ and ϵ. We can see that for large δ the speedup factor declines since δ determines T and increasing δ decreases the relative weight of τ.

Figure 12 compares the results of time-limited and time-unlimited cases. The graphs for the two cases are very similar, meaning that the overhead of requiring a guaranteed ($\delta = 0.0$) ϵ-correct solution is very low. Note that only the optimal method is able to handle the time-unlimited case.

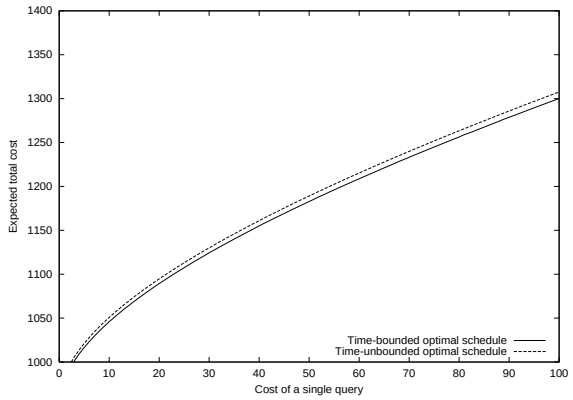


Figure 12: The expected total cost as a function of τ for the time-limited and time-unlimited cases.

6.3 Normal distribution

In this section we first show some formal properties of optimal schedules for normal distribution, supply a numerical solution for the communication problem presented in the introduction and present simulation results.

6.3.1 Formal solution

The normal distribution with mean value m and deviation σ is described by the density function

$$f(t) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(t-m)^2}{2\sigma^2}}, \tag{48}$$

and its distribution function is

$$F(t) = \frac{1}{\sqrt{2\pi}\sigma} \int_{-\infty}^{t} e^{-\frac{(x-m)^2}{2\sigma^2}} dx. \tag{49}$$

Since we use $t_0 = 0$, we should have used a truncated normal distribution with a distribution density

$$\frac{1}{(1-\mu)} \cdot \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(t-m)^2}{2\sigma^2}},$$

and a distribution function

$$\frac{1}{1-\mu} \cdot \left[\frac{1}{\sqrt{2\pi}\sigma} \int_{-\infty}^{t} e^{-\frac{(x-m)^2}{2\sigma^2}} dx - \mu \right],$$

where

$$\mu = \frac{1}{\sqrt{2\pi}\sigma} \int_{-\infty}^{0} e^{-\frac{(x-m)^2}{2\sigma^2}} dx.$$

In the following experiments, m is large enough to allow us to neglect μ and use a standard normal distribution. We now prove the following proposition about the behavior of time sequences from $\mathcal{W}$ for normal distribution.

Proposition 5 *Let $\mathcal{T} = \langle t_1, \ldots, t_n \rangle$ be a sequence from $\mathcal{W}$. Let $\Delta_i = t_i - t_{i-1}$. If ζ is normally distributed with mean value m and standard deviation σ, and $u(t) = t$, then the following inequalities hold:*

- *If $max(t_{i-1}, t_i) < m$, then $\Delta_i < \Delta_{i-1}$.*
- *If $\Delta_i > \Delta_{i-1}$ and $t_i > t_{i-1} > m$, then $\Delta_{i+1} > \Delta_i$.*

These conditions mean that the intervals between queries form a decreasing sequence up to some point $\tilde{t} \geq m$, and an increasing sequence after this point.

Proof: The first inequality follows from the fact that $f(t)$ is an increasing function for $t < m$ and (21). From (21) we obtain that there are such points ξ between t_{i-1} and t_i, and η between t_{i-1} and t_{i-2} such that:

$$\Delta_{i+1} - \Delta_i = \frac{F_i - F_{i-1}}{F'_i} - \frac{F_{i-1} - F_{i-2}}{F'_{i-1}} = \frac{F'(\xi)}{F'_i}\Delta_i - \frac{F'(\eta)}{F'_{i-1}}\Delta_{i-1}.$$

Since $t_i > t_{i-1} > m$, we have:

- From $\eta < \xi$, we have that $F'(\eta) > F'(\xi)$.
- From $t_i > t_{i-1}$, we have that $F'_i < F'_{i-1}$.

Therefore,

$$\Delta_{i+1} - \Delta_i > \frac{F'(\xi)}{F'_{i-1}}\Delta_i - \frac{F'(\xi)}{F'_{i-1}}\Delta_{i-1} = \frac{F'(\xi)}{F'_{i-1}}(\Delta_i - \Delta_{i-1}) > 0.$$

□

6.3.2 Communication example

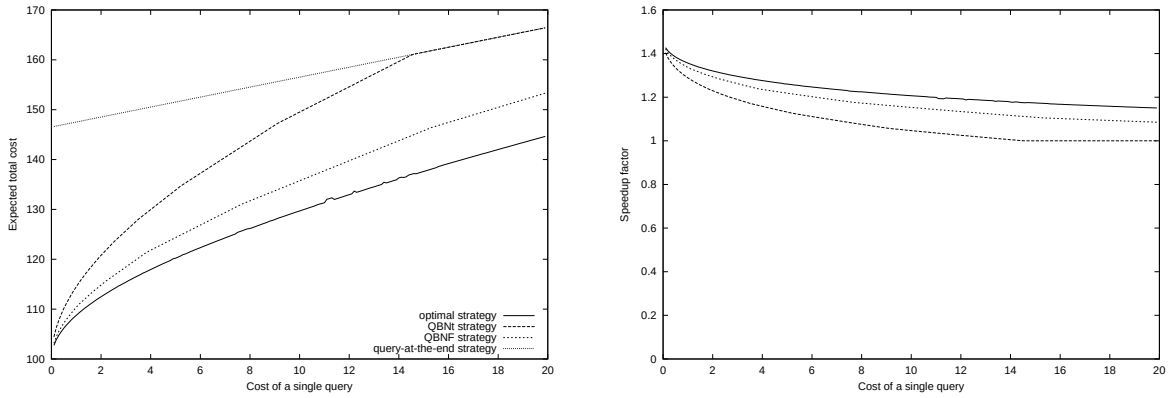
Suppose that two stations A and B want to communicate with each other with the help of a receiver-transmitter robot C, as was described in the introduction (see Figure 1). We can assume that the probability of the robot to reach point D before time t is distributed by a truncated normal law with mean value m and standard deviation σ. We assume that the distribution parameters are known either by theoretical analysis or from previous experience. Let τ be the cost of a single communication attempt. We assume that we are either given a time limit T or a desired success probability $1 - \delta$. In the second case we compute T from the formula

$$1 - \frac{1}{\sqrt{2\pi}\sigma}\int_0^T e^{-\frac{(x-T)^2}{2\sigma^2}}dx = \delta. \tag{50}$$

If no error is permitted, i.e., $\delta = 0$, we have the case of a time-unlimited problem. Our goal is to design a schedule with a minimal expected time until communication is established.

6.3.3 Simulation results

We conducted a set of simulation experiments with m, the average time of moving the robot to point D, set to 100. σ was set to 20. δ was set to 0.01, which corresponds to $T = 146.53$. We tested the effect of the independent variable τ on the expected total cost. The test was run with four different scheduling strategies: the optimal algorithm, the QBN_t strategy, the



(a) (b)

Figure 13: (a) Normal distribution: The expected total cost as a function of the cost of a single query for various scheduling strategies. (b) Normal distribution: The speedup factor of the three more sophisticated methods relative to the query-at-the-end method as a function of τ.

QBN_F strategy and the query-at-the-end strategy. τ was varied between 0.1 and 20 with a default value of 10.

The results are shown in Figure 13.

Here we can see an advantage of the optimal method over the other three methods. We can also observe an advantage of the QBN_F method over the QBN_t method, since the first yields schedules with even areas.

We also conducted an experiment with $T = \infty$; the results were identical to those obtained above for the time-limited case. Note that the other three methods are not able to handle the time-unlimited case.

7 Experimentation with a real problem

In the previous section we showed experimental results for some simulated data. Here we test our framework on a real-world problem where the distribution function is not externally supplied.

Assume that our task is to generate hard solvable search problems for a given search space. A problem is defined by a pair of states, an initial state and a goal state. Assume that we possess a heuristic function which estimates the cost of the shortest path between two states. Assume that we define a *hard* problem as a pair of states with a heuristic distance of at least k. If the operators are reversible, one of the possible methods for generating such problems is to generate a random goal state and perform a random walk from the goal state. After each operator application we check the current heuristic distance. When hitting a state with distance equal or greater than k we stop the process.

While the above approach is intuitive, it may carry unnecessarily high costs if the heuristic function is expensive. We will show here how monitoring can be used to make the generation process more efficient. A monitoring schedule in this context specifies the number of operator applications between successive calls to the heuristic function. While generating problems, we learn the distribution function F of the number of steps required for achieving the goal.

When sufficient data is accumulated, we start applying the monitoring algorithm to design optimal schedules.

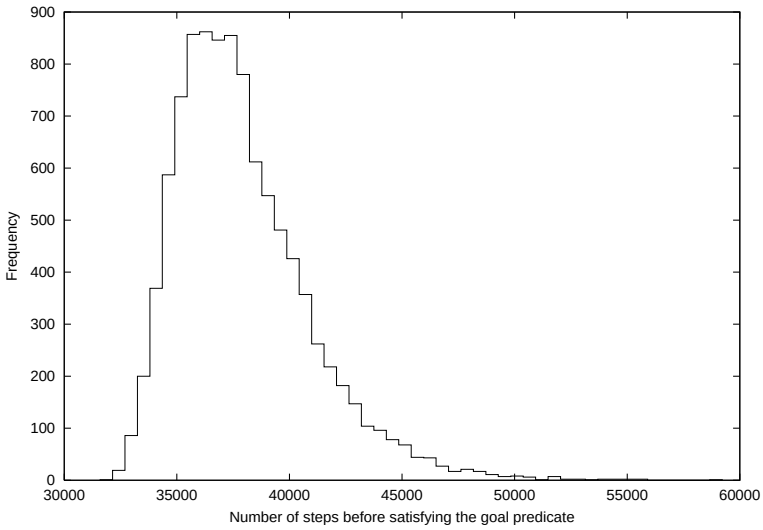


Figure 14: A histogram describing the distribution function estimated for the puzzle domain based on past problems.

We implemented this approach for generating problems from the $N \times N$ sliding-tile puzzle domain, using the sum-of-Manhattan-distances as our heuristic estimator. The cost of applying the heuristic is $O(N^2)$ while applying a single operator is $O(1)$. We have run an experiment testing the algorithm for 100×100 puzzles and a required threshold distance of $10,000$. We used the first 10,000 problems for estimating the distribution function and obtained the histogram shown in Figure 14. We then applied our algorithm and obtained a schedule. We tested the resulting schedule by generating another 10,000 problems. Table 1 shows the results obtained. For comparison, we added the results for the strategies described in Section 2.

Scheduling method	Total cost	Standard deviation
query-at-the-end	58,946	0
query-every-Δt	75,891	6,076
QBN_t	44,373	8,600
QBN_F	48,799	10,886
Optimal	43,511	4,824

Table 1: The results obtained for generating hard 100×100 problems using various scheduling methods.

As can be seen from the table, the optimal strategy performs better than all the other approaches. Note that the above problem does not satisfy one of our assumptions – that

the binary quality measurement should be monotonic. Nevertheless, our method was able to handle the problem quite nicely.

8 Related work

In this section we compare our method to existing works. In Section 8.1 we discuss a monitoring scheme designed by Russell and Wefald [22], and in Section 8.2 a framework described by Hansen and Zilberstein [9].

8.1 Fixed-step monitoring in $DTA*$

Russell and Wefald [22] describe a search algorithm $DTA*$ implementing a decision-theoretic control method. The algorithm finds the next node to be expanded by estimating the potential information gain expected by expanding this node. This estimation is performed by an anytime local search procedure which continues to run as long as it is expected to be beneficial. This test for the benefit of continuation is analogous to our query for the satisfaction of the goal predicate. Instead of performing the test after each step of the procedure, Russell and Wefald propose that the test be performed each *grain-size* steps denoted[14] by G, thus reducing the number of times that the test is performed. The average number of node expansions per search is denoted by A, the average number of tests by N ($N = A/G$), and the ratio of the cost of a test to the cost of a node expansion by ρ. The authors also make an assumption that half of the final G node expansions are wasted. They prove that the optimal grain-size is $\sqrt{2\rho A}$.

This scheme corresponds to the QBN_t strategy presented in Section 2.3. Russell and Wefald do not explicitly make an assumption about the type of distribution involved. Their assumption on $G/2$ wasted nodes, however, indicates that they assume uniform distribution. This is a reasonable assumption for search problems of the type they deal with. The framework of Russell and Wefald can be generalized to general distribution (with certain constraints) using the following theorem.

Theorem 12 *Let F be a distribution function, $u(t) = t$ a cost function, T a maximal allocated time, τ a time required for a single query, and p a probability of the algorithm's success. Let us define $\bar{F} = 1/(n-1)\sum_{i=1}^{n-1} F(iT/n)$. If we assume that $\bar{F}$ is independent of n (or that its dependency on n can be neglected), then the optimal number of queries for the QBN_t strategy can be written approximately[15] as*

$$n_{opt} = \sqrt{\frac{p\bar{F}}{1-p\bar{F}}\frac{T}{\tau}}. \tag{51}$$

Proof: A schedule $\mathcal{T} = \langle t_1, \ldots, t_n \rangle$ with equal intervals between time points meets $t_i = \frac{i}{n} t_n$. By definition of equal-step schedule, $t_n = T$. Substituting these values for t_i in (4) and using the fact that $C = 0$ (see Proposition 3), we obtain

$$E(\mathcal{T}) = (T + n\tau) - p\sum_{i=1}^{n-1}\left(\frac{T}{n} + \tau\right) F\left(\frac{i}{n}T\right) = (T + n\tau)\left(1 - p\frac{n-1}{n}\bar{F}\right).$$

[14] We use the notation used by the authors.
[15] Due to the possible discretization error.

Opening the parentheses, we obtain

$$E(\mathcal{T}) = (T + n\tau)(1 - p\bar{F} + p\bar{F}/n) = \frac{T}{n}p\bar{F} + n\tau(1 - p\bar{F}) + T - p\bar{F}(T - \tau). \qquad (52)$$

As in Section 6.1, we perform minimization of the above expression by n, assuming that n is continuous. Since $\bar{F}$ is independent of n, the necessary condition for local extremum has the form

$$\frac{dE(\mathcal{T})}{dn} = -\frac{T}{n^2}p\bar{F} + \tau(1 - p\bar{F}) = 0,$$

yielding the single solution expressed by (51). Since the second derivative of $E(\mathcal{T})$ by n is strictly positive, $E(\mathcal{T})$ is convex, and therefore the found solution is a global minimum. □

The assumption about the invariant $\bar{F}$ holds automatically for uniform distribution because

$$\bar{F} = \frac{1}{n-1}\sum_{i=1}^{n-1} F\left(\frac{iT}{n}\right) = \frac{1}{n-1}\sum_{i=1}^{n-1}\frac{i}{n} = \frac{1}{n-1}\cdot\frac{n(n-1)}{2n} = \frac{1}{2}.$$

The assumption holds for large n according to the law of large numbers. Unfortunately, small values of n for other distributions can violate this assumption.

In terms of the $DTA*$ algorithm, the grain size G can be viewed as T/n in our notation, which by definition represents an interval between queries. The ratio parameter ρ corresponds to τ, if we measure time by expanded nodes. We can show that the average number of node expansions per search, A, can be approximated by $(1 - p\bar{F})T$. For the uniform distribution this approximation is correct because $F(t) = t$ and $\bar{F} = 1/2$. For other distributions, we can show the correctness for large ns since an *approximate* value for A can be expressed by the formula

$$A \approx p\sum_{i=1}^{n} t_i(F(t_i) - F(t_{i-1})) + (1 - pF(T))T =$$

$$p\sum_{i=1}^{n}\frac{i}{n}T(F(t_i) - F(t_{i-1})) + (1 - pF(T))T = p\frac{T}{n}(nF(T) - \sum_{i=1}^{n-1} F_{t_i}) + (1 - pF(T))T =$$

$$T\left(1 - p\frac{n-1}{n}\bar{F}\right) = T(1 - p\bar{F}) + T\frac{p\bar{F}}{n} \approx T(1 - p\bar{F}).$$

Similarly, the average number of tests, N, is approximately $(1 - p\bar{F})n$. Substituting these values in the right part of (52) gives us $p\bar{F}G + \rho\frac{A}{G}$, which is the formula used in [22] with $p\bar{F} = 1/2$. In the particular case described by Russell and Wefald, we can assume that the nodes are uniformly distributed and $p = 1$, and therefore $\bar{F} = 1/2$, $A = T/2$, $N = n/2$ and the optimal number of queries (which is also an optimal grain size) will be

$$G_{opt} = \frac{T}{n_{opt}} = \sqrt{T\tau} = \sqrt{2\rho A}.$$

This is the result presented by Russell and Wefald.

Theorem 12 implies the following two corollaries.

Corollary 3 *If a distribution satisfies the conditions of Theorem 12, and its density is skewed towards 0, then $\bar{F}$ is close to 1, and n_{opt} is large. If the density is skewed towards T, then $\bar{F}$ is close to 0, and n_{opt} is small.*

This corollary formalizes our intuition that it is better to ask more often in the case that the monitored event is likely to occur early, and to ask at the end in the case that it is likely to occur late.

Corollary 4 *If the cost function is $u(t) = t$, then for the uniform distribution the expected elapsed time for the QBN_t strategy is*

$$E(t_1, \ldots, t_n) = \left(1 - \frac{p}{2} + \frac{p}{2n}\right)(T + n\tau), \tag{53}$$

and the optimal number of queries is

$$n_{opt} = \sqrt{\frac{p}{2-p}\frac{T}{\tau}}. \tag{54}$$

Proof: Substituting $\bar{F} = 1/2$ to (52), we obtain

$$E(\mathcal{T}) = \frac{T}{2n}p + n\tau\left(1 - \frac{p}{2}\right) + T - \frac{p}{2}(T - \tau).$$

Simplifying this expression, we get (53). The expression for n_{opt} is obtained from (51) by substituting $1/2$ for $\bar{F}$. □

Note that the equations above resemble those for the optimal schedule given in Section 6.1.

We have repeated the experiments described in the previous section using the above method. In cases where the above assumptions did not hold, we used an exhaustive optimization over n. The results for the uniform distribution were only slightly worse than the results obtained by the optimal scheme. For exponential distribution, this strategy was as good as the optimal. For normal distribution, however, the optimal strategy outperformed the QBN_t strategy by about 30%.

8.2 Dynamic monitoring

Zilberstein and Hansen [8, 9] proposed a monitoring strategy based on dynamic programming. In this subsection we prove that, under unifying assumptions, the optimal schedules obtained by their method are equivalent to those generated by ours. In the analysis below we adopt the definitions used by Zilberstein and Hansen. We will refer to their method as the *dynamic* method and to ours as the *static* method.

The dynamic method looks for a *monitoring policy* which, at each quality level q_i and time step t_k, provides a monitoring decision $\Delta t, m$ where Δt represents the additional amount of time to allocate to the anytime algorithm, and m is a binary variable that stands for the two options after Δt: perform monitoring or stop [8].

An optimal monitoring policy is found by dynamic programming methods using the rule

$$V(q_i, t_k) = \max_{\Delta t, m} \begin{cases} \sum_j \left(Pr(q_j|q_i, \Delta t)U(q_j, t_k + \Delta t)\right), & \text{if } m = stop \\ \sum_j \left(Pr(q_j|q_i, \Delta t)V(q_j, t_k + \Delta t)\right) - C, & \text{if } m = monitor, \end{cases} \tag{55}$$

where t_k is a query time point, q_j is a quality level, $\Delta t, m$ is a monitoring decision, $U(q, t)$ is a utility function, C is a query cost, and $V(q, t)$ is a value function being optimized. $Pr(q_j|q_i, \Delta t)$ is the conditional probability of getting a solution of quality q_j by running the

algorithm for additional Δt time, when the current solution has quality q_i. This probability is called the *dynamic performance profile* of the algorithm.

To facilitate the comparison between the two methods, we transform (55) to the following equivalent form:

$$V(q_i, t_k) = \max_{\Delta t, m} \begin{cases} U(q_i, t_k) + C, & \text{if } m = stop \\ \sum_j \left(Pr(q_j|q_i, \Delta t)V(q_j, t_k + \Delta t)\right) - C, & \text{if } m = monitor. \end{cases} \quad (56)$$

This form assumes that the monitoring decision refers to the *current* step and not to the *next* one. An additional term C for the stop action is intended to compensate for the extra call for the monitoring procedure.

There are two main differences between this model and ours:

- The dynamic method is Markovian and predicts quality information based on the current quality level and the allocated time. The static method, on the other hand, assumes that the quality depends only on the time spent by the process. This is the reason why we use probabilistic performance profiles rather than dynamic performance profiles.
- We assume that a query has two types of associated costs: some resource cost C and time cost τ, while Zilberstein and Hansen assume only a resource cost C. Therefore, a query in our model delays the finishing time of the process while in the dynamic method it does not.

We now want to rewrite (56) in the terms of our model described by (3):

1. Since the dynamic model does not support the probability of success p and the query delay τ, we assume $p = 1$, and $\tau = 0$. T in the static model is equivalent to t_n in the dynamic one.
2. In our model the goal predicate is Boolean, and therefore it partitions the set of states to two equivalence classes. Without loss of generality, we can therefore assume that only two states are available, namely q_0 and q_1, where q_1 is the final state, and q_0 is not. We also assume that once the process reaches q_1, it cannot leave it. This requirement is natural for anytime algorithms.
3. We assume that monitoring the process at t_k does not add any information unless the goal predicate is satisfied. Thus t fully determines the prediction. In the dynamic model, on the other hand, the prediction is determined by the previous state and Δt. To unify the two approaches we use the extension of dynamic performance profile $Pr(q_j|q_i, t_k, \Delta t)$, which stands for the probability that quality q_j will be obtained at $t_k + \Delta t$ given that at t_k the quality level was q_i. We can now replace $Pr(q_j|q_i, \Delta t)$ in (56) with its extended form.
4. We define the utility function $U(q, t)$ as follows:

$$U(q_i, t) = \begin{cases} -\infty, & \text{if } i = 0 \text{ and } t < T \\ -u(t), & \text{if } i = 1 \text{ or } t \geq T. \end{cases}$$

This means that no further monitoring is required after either the goal predicate is satisfied or the time limit T is exceeded, and the utility is the opposite of the cost.

Let us denote by $Pr(q_i, t)$ the probability that quality q_i has been achieved at t. In terms of our model

$$\begin{aligned} Pr(q_0, t) &= 1 - F(t) \\ Pr(q_1, t) &= F(t). \end{aligned}$$

Since the process cannot leave the goal state,

$$\left\{ \begin{array}{lcl} Pr(q_0|q_1, t_k, \Delta t) &=& 0 \\ Pr(q_1|q_1, t_k, \Delta t) &=& 1. \end{array} \right. \tag{57}$$

This means that in formula (56), for $i = 1$ (goal state achieved) only $j = 1$ has a non-zero contribution to the sum, and therefore $V(q_1, t_k)$ has the following form:

$$V(q_1, t_k) = \max_{\Delta t, m} \left\{ \begin{array}{ll} U(q_1, t_k) + C, & \text{if } m = stop \\ V(q_1, t_k + \Delta t) - C, & \text{if } m = monitor. \end{array} \right. \tag{58}$$

Since $C \geq 0$ and $U(q_1, t) = -u(t)$ is a non-increasing function by t, $V(q_1, t)$ is also non-increasing by t. Therefore, in this case the optimal solution will be to set $\Delta t = 0$ and stop immediately.

In the case where the goal is not achieved, $V(q_0, t_k)$ has the form

$$V(q_0, t_k) = \max_{\Delta t, m} \left\{ \begin{array}{ll} U(q_0, t_k), & \text{if } m = stop \\ Pr(q_0|q_0, t_k, \Delta t)V(q_0, t_k + \Delta t) + & \\ Pr(q_1|q_0, t_k, \Delta t)V(q_1, t_k + \Delta t) - C, & \text{if } m = monitor. \end{array} \right. \tag{59}$$

If $t_k \geq T$, then $U(q_i, t_k) = -u(t_k)$, and the process should be stopped immediately for the same reasons as above. If $t_k < T$, then $U(q_0, t) = -\infty$, and only the monitoring option is available. Therefore,

$$V(q_0, t_k) = Pr(q_0|q_0, t_k, \Delta t)V(q_0, t_k + \Delta t) + Pr(q_1|q_0, t_k, \Delta t)V(q_1, t_k + \Delta t) - C. \tag{60}$$

$Pr(q_0|q_0, t_k, \Delta t)$ stands for the probability that the process will remain at state q_0 for Δt time, given that q_0 was observed at t_k. Therefore, by the definition of conditional probability, we can write that

$$Pr(q_0|q_0, t_k, \Delta t) = \frac{1 - F(t_k + \Delta t)}{1 - F(t_k)}.$$

Similarly, $Pr(q_1|q_0, t_k, \Delta t)$ stands for the probability that the process will switch to state q_1 within Δt time, given that q_0 was observed at t_k.

$$Pr(q_1|q_0, t_k, \Delta t) = \frac{F(t_k + \Delta t) - F(t_k)}{1 - F(t_k)}.$$

Thus,

$$\begin{aligned} V(q_0, t_k) &= \frac{1 - F(t_k + \Delta t)}{1 - F(t_k)} V(q_0, t_k + \Delta t) + \frac{F(t_k + \Delta t) - F(t_k)}{1 - F(t_k)} V(q_1, t_k + \Delta t) - C = \\ & \frac{1 - F(t_k + \Delta t)}{1 - F(t_k)} V(q_0, t_{k+1}) + \frac{F(t_{k+1}) - F(t_k)}{1 - F(t_k)} (U(q_1, t_k) + C) - C = \\ & \frac{1 - F(t_{k+1})}{1 - F(t_k)} V(q_0, t_{k+1}) + \frac{F(t_{k+1}) - F(t_k)}{1 - F(t_k)} U(q_1, t_k) - \frac{1 - F(t_{k+1})}{1 - F(t_k)} C. \end{aligned} \tag{61}$$

Let us denote $F(t_k)$ by F_k, and $U(q_1, t_k)$ by U_k. Computing eqeq:Vqt for $k = 0$ gives us

$$V(q_0, t_0) = (1 - F_1)V(q_0, t_1) + (F_1 - F_0)U_1 - (1 - F_1)C.$$

Substituting $V(q_0, t_1)$ in the above formula using (61) yields

$$V(q_0, t_0) = (1 - F_2)V(q_0, t_2) + (F_2 - F_1)U_2 + (F_1 - F_0)U_1 - (1 - F_2)C - (1 - F_1)C.$$

For inductive reasons, we can write the general case as

$$V(q_0, t_0) = (1 - F_n)V(q_0, t_n) + \sum_{i=1}^{n}(F_i - F_{i-1})U_i - \sum_{i=1}^{n}(1 - F_i)C. \tag{62}$$

Assuming that the process stops after n steps, we have $V(q_0, t_n) = U(t_n)+C$, and therefore the value function can be rewritten as

$$\begin{aligned} V(q_0, t_0) &= (1 - F_n)(U_n + C) + \sum_{i=1}^{n}(F_i - F_{i-1})U_i - \sum_{i=1}^{n}(1 - F_i)C = \\ &\quad (1 - F_n)U_n + \sum_{i=1}^{n}(F_i - F_{i-1})U_i - \sum_{i=1}^{n}(1 - F_{i-1})C. \end{aligned}$$

On the other hand, we can see that

$$\sum_{i=1}^{n}(1 - F_{i-1})C = nC - \sum_{i=1}^{n} F_{i-1} = nC(1 - F_n) + C\sum_{i=1}^{n} i(F_i - F_{i-1}),$$

and therefore

$$\begin{aligned} V(q_0, t_0) &= (1 - F_n)U_n + \sum_{i=1}^{n}(F_i - F_{i-1})U_i - \sum_{i=1}^{n}(F_i - F_{i-1})C - nC + nF_n = \\ &\quad \sum_{i=1}^{n}(F_i - F_{i-1})(U_i - C) + (1 - F_n)(U_n - nC) = \\ &\quad \sum_{i=1}^{n}(F(t_i) - F(t_{i-1}))(-u(t_i) - C) + (1 - F(t_n))(-u(t_n) - nC). \end{aligned}$$

Finally, using (3) we obtain

$$V(q_0, t_0) = -E(t_1, \ldots, t_n). \tag{63}$$

This proves that the two models are equivalent under the unifying assumptions. The dynamic model has the advantage of being able to effectively use additional information obtained from monitoring, but has the disadvantage of requiring rich statistical data. It also has high complexity due to the dynamic programming involved. If we denote the size of the set of possible time values by $|t|$, then the complexity will be $O(|t|^2)$. The static model has the advantage of efficiency because its only time-consuming module is minimization of a one-variable function. Even if we had used a dumb minimization algorithm that tests the whole set of time points, the static method would still require fewer operations. This is because a test of each point requires a number of operations proportional to the schedule size, which is much smaller than $|t|$. In practice, for smooth distribution functions, minimization methods are very cheap.

Another advantage of the static method over the dynamic one is its ability to handle queries that consume time from the monitored process. This is significant when $u(t) \neq t$. It looks as if each of the methods has its advantages, and it would be interesting to somehow combine the two.

9 Discussion

This work studies the problem of monitoring anytime processes. We took the approach of attacking a limited but well-defined monitoring problem and performing a thorough theoretical study of its solution. We assume that the anytime process can be queried for a binary goal condition, and that processing the query consumes time from the monitored process. Our goal is to design an optimal query schedule. Querying too often significantly increases the overhead and delays the time it takes for the goal predicate to be satisfied. Querying too infrequently, on the other hand, delays the time it takes to detect the satisfaction of the goal predicate.

We introduce an algorithm for generating optimal schedules and prove its properties. The algorithm is based on an inductive formula that builds an entire optimal schedule based on the time point of its first query. This method reduces the problem to one-variable optimization, which is a well-studied problem with many analytical and numerical solutions.

After describing our general algorithm and proving its properties, we perform distribution-based analysis of the problem and conduct a set of experiments on simulated data to confirm our analysis. The experimental results show the advantage of our optimal method over the default monitoring strategy of querying once at the end of the process. We also achieved positive results applying the algorithm on a real problem.

Our approach requires a probability distribution over when the goal predicate is satisfied. This requirement is weaker than the need for the dynamic performance profile assumed by most works on monitoring anytime algorithms (this stronger requirement allows on-line monitoring with better expected results). There are several possible ways for obtaining such information in practice. One possibility is that the distribution function is externally supplied. A more likely case is that the distribution type is known and only the parameters need to be estimated. Another possibility is that the algorithm being monitored solves a sequence of similar problems (such an assumption is commonly used in the COLT community). In such a case we can use the past experience to approximate the distribution function. Our experiments show that while the schedules produced using the approximated distribution function may not be optimal, they still provide significant gains.

Minimizing the expected cost is not the only possible goal. Sometimes minimizing the standard deviation (risk) is desirable as well[16]. Allowing a tradeoff between the expected total cost and the variance in the monitoring context is an interesting and open research problem.

We believe that this work contributes both to the foundations of meta-reasoning in general and monitoring in particular. It does so by providing a rigorous mathematical analysis of the problem and its solution, along with an applicable algorithm which can be applied to many real problems.

References

[1] A. Blumerand, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM*, 36(4):929–965, 1989.

[16] Huberman *et al.* [15], for example, investigated such a setup where the goal is to reduce both the expected cost and the variance over either solving different instances of a problem or even running multiple trials of solving the same instance.

[2] M. Boddy and T. Dean. Decision-theoretic deliberation scheduling for problem solving in time-constrained environments. *Artificial Intelligence*, 67(2):245–286, 1994.

[3] John S. Breese and Eric. J. Horvitz. Ideal reformulation of belief networks. In *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence*, pages 129–143, 1991.

[4] R. P. Brent. *Algorithms for Minimization without Derivatives*. Prentice Hall, Englewood Cliffs, New Jersey, 1973.

[5] T. Dean and M. Boddy. An analysis of time-dependent planning. In *Proceedings of Seventh National Conference on Artificial Intelligence*, pages 49–54, Minneapolis, Minnesota, 1988.

[6] J. Fuernkranz. Integrative windowing. *Journal of Artificial Intelligence Research*, 8:129–164, 1998.

[7] I. J. Good. Twenty-seven principles of rationality. In V.P. Godambe and D.A. Sprott, editors, *Foundations of Statistical Inference*, pages 108–141. Holt, Rinehart, Winston, Toronto, 1971.

[8] E. A. Hansen and S. Zilberstein. Monitoring the progress of anytime problem-solving. In *Proceedings of the 13th National Conference on Artificial Intelligence*, pages 1229–1234, Portland, Oregon, 1996.

[9] E. A. Hansen and S. Zilberstein. Monitoring anytime algorithms. *SIGART Bulletin*, 7(2), 1997.

[10] E. J. Horvitz. Reasoning about beliefs and actions under computational resource constraints. In *Proceedings of the 1987 Workshop on Uncertainty in Artificial Intelligence*, Seattle, Washington, 1987.

[11] E. J. Horvitz. *Computation and Action under Bounded Resources*. PhD thesis, Stanford University, 1990.

[12] E. J. Horvitz. Models of continual computation. In *Proceedings of the 14th National Conference on Artificial Intelligence*, 1997.

[13] E. J. Horvitz, C. F. Cooper, and D. E. Heckerman. Reflection and action under scare resources: Theoretical principles and empirical study. In *Proceedings of the Eleventh International Conference on Artificial Intelligence*, pages 1121–1127, 1989.

[14] Eric Horvitz and Adrian Klein. Reasoning, metareasoning, and mathematical truth: Studies of theorem proving under limited resources. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, Montreal, August 1995.

[15] Bernardo A. Huberman, Rajan M. Lukose, and Tad Hogg. An economic approach to hard computational problems. *Science*, 275:51–53, January 1997.

[16] Michael J. Kearns and Umesh V.Vazirani. *An Introduction to Computational Learning Theory*. The MIT Press, Cambridge, Massachusetts, 1994.

[17] O. Ledeniov and S. Markovitch. The divide-and-conquer subgoal-ordering algorithm for speeding up logic inference. *Journal of Artificial Intelligence Research*, 9:37–97, 1998.

[18] O. Ledeniov and S. Markovitch. Learning investment functions for controlling the utility of control knowledge. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 463–468, Madison, Wisconsin, 1998.

[19] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge University Press, 1993.

[20] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[21] S. Russell and E. Wefald. On optimal game-tree search using rational metareasoning. In *Proceedings of the Eleventh International Conference on Artificial Intelligence*, pages 334–340, 1989.

[22] S. Russell and E. Wefald. *Do the Right Thing: Studies in Limited Rationality.* The MIT Press, Cambridge, Massachusetts, 1991.

[23] S. J. Russell. Rationality and intelligence. *Artificial Intelligence*, 4:55–77, 1997.

[24] S. J. Russell and S. Zilberstein. Composing real-time systems. In *Proceedings of the Twelfth National Joint Conference on Artificial Intelligence (IJCAI-91)*, Sydney, 1991. Morgan Kaufmann.

[25] Herbert A. Simon. A behavioral model of rational choice. *Quarterly Journal of Economics*, 69:99–118, 1955.

[26] V. N. Vapnik and A. Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264–280, 1971.

[27] S. Zilberstein. Operational rationality through compilation of anytime algorithms. *Ph.D.Dissertation, Computer Science Division, University of California, Berkeley*, 1993.

[28] S. Zilberstein and S. J. Russell. Efficient resource-bounded reasoning in AT-RALPH. In *Proceedings of the First International Conference on AI Planning Systems*, pages 260–266, College Park, Maryland, 1992.

Appendix A: Weakening the smoothness assumptions

Assume that we are given a time cost function $u(t)$ and a distribution function $F(t)$. From the definition, both $u(t)$ and $F(t)$ are non-decreasing functions. Assume also that both $u(t)$ and $F(t)$ have a first derivative over $\mathcal{R}$, with the possible exception of a countable number of discontinuity points of the first type, i.e., points where $u(t-0) \neq u(t+0)$. Let us suppose also that the number of intervals where $u(t)$ is constant is also countable. These requirements are quite natural for real problems.

Our goal is to construct differentiable functions $\tilde{u}(t)$ and $\tilde{F}(t)$, which will be as close as desired to $u(t)$ and $F(t)$. In addition, $\tilde{u}(t)$ must be a monotonic increasing function. We will show the smoothing process for $u(t)$. Smoothing of $F(t)$ is performed in a similar way.

Let $\{x_1, \ldots, x_m, \ldots\}$ be a set of discontinuity points of $u(t)$. Let ϵ be an arbitrarily small number. Let us define a new utility function $\tilde{u}(t)$ in the following way:

1. $\tilde{u}(t) = u(t)$ when t is not within the ϵ-neighborhood of any of discontinuity points, i.e., $t \notin \bigcup_i (x_i - \epsilon, x_i + \epsilon)$.

2. In the ϵ-neighborhood we define $\tilde{u}(t)$ as a smooth increasing function, such that $\tilde{u}(t-\epsilon) = u(t-\epsilon)$ and $\tilde{u}(t+\epsilon) = u(t+\epsilon)$.

The resulting function will be smooth enough and will differ from $u(t)$ in an arbitrarily small set of points. It is easy to see that the effect of smoothing on the expected cost can be made arbitrarily small.

We also want $u(t)$ to be monotonically increasing. Suppose that $u(t)$ is smoothed already using the procedure above. Let $u(t)$ be constant on the intervals (x_k, x_{k+1}). Let ϵ and δ be arbitrarily small numbers. As before, we define $\tilde{u}(t)$ to be equal to $u(t)$ outside the ϵ-neighborhoods of these intervals, and a smooth increasing function in $(x_k - \epsilon, x_{k+1} + \epsilon)$, such that $\tilde{u}(x_k - \epsilon) = u(x_k) - \delta$ and $\tilde{u}(x_k + \epsilon) = u(x_k) + \delta$. As before, $\tilde{u}(t)$ can be constructed to be arbitrarily close to $u(t)$.

Appendix B: Formal proofs

Proof of Theorem 9

First we want to prove (32). For a uniform distribution, equation (15) has the form

$$t_{i+1} - 2t_i + t_{i-1} = \tau \text{ for } i = 1, \ldots, n-1. \tag{64}$$

Since $t_0 = 0$ and $t_n = T$, we have a tridiagonal system of equations. We can easily show by induction on i that

$$t_i = i\left(\frac{1}{i+1}t_{i+1} + \frac{1}{2}\tau\right) \text{ for } i = 1, \ldots, n-1. \tag{65}$$

Indeed, for $t_0 = 0$

$$-t_0 + 2t_1 - t_2 = \tau \Rightarrow t_1 = \frac{1}{2}t_2 + \frac{1}{2}\tau.$$

If we suppose that (65) holds for $i = k < n-1$, then for $i = k+1$ we get the following expression:

$$\begin{aligned} &-t_k + 2t_{k+1} - t_{k+2} = \tau \;\Rightarrow \\ &\quad \Rightarrow \frac{k+2}{k+1}t_{k+1} - \frac{k}{2}\tau - t_{k+2} = \tau \Rightarrow t_{k+1} = (k+1)\left(\frac{1}{k+2}t_{k+2} + \frac{1}{2}\tau\right), \end{aligned}$$

which finishes the proof.

We will now show the correctness of formula (32) using descending induction. The base of the induction is derived immediately from (65) with $i = n-1$. If we assume that (32) holds for $i = k > 1$, then for $i = k-1$ we get

$$\begin{aligned} t_k &= k\left(\frac{T}{n} + \frac{n-k}{2}\tau\right) \Rightarrow \\ t_{k-1} &= (k-1)\left(\frac{t_k}{k} + \frac{\tau}{2}\right) = (k-1)\left(\frac{T}{n} + \frac{n-(k-1)}{2}\tau\right), \end{aligned}$$

which proves (32).

Since $t_i \le T$, by (32) we have

$$i\left(\frac{T}{n}+\frac{n-i}{2}\tau\right) \le T \Rightarrow i\frac{n-i}{2}\tau \le \frac{n-i}{n}T \Rightarrow \frac{T}{\tau} \ge \frac{ni}{2} \Rightarrow \frac{T}{\tau} \ge \frac{n(n-1)}{2}. \tag{66}$$

Due to Proposition 1, the last inequality is the necessary and sufficient condition for the time sequence to be increasing. The restriction of (33) follows immediately from the solution. If n is too large, the minimized function will obtain its minimum only on the border, i.e., there exists i such that $t_i = t_{i+1}$, leading to redundant queries.

Now we want to find the value of $E(\mathcal{T})$ where the time points t_i satisfy (32). We can see that for a uniform distribution,

$$E(t_1,\dots,t_n) = \frac{p}{T}\sum_{i=1}^{n}(t_i+i\tau)(t_i-t_{i-1}) + (1-p)(T+n\tau). \tag{67}$$

Substituting the values for t_i defined by (32) in (67), we get

$$\begin{aligned}
&E(t_1,\dots,t_n) = \\
&\quad = \frac{p}{T}\sum_{i=1}^{n}\left(\frac{i}{n}T+\frac{i(n-i)}{2}\tau+i\tau\right)\times \\
&\quad \times\left(\frac{i}{n}T+\frac{i(n-i)}{2}\tau-\frac{i-1}{n}T-\frac{(i-1)(n-(i-1))}{2}\tau\right)+(1-p)(T+n\tau) = \\
&\quad = \frac{p}{T}\sum_{i=1}^{n} i\left(\frac{T}{n}+\frac{n-i+2}{2}\tau\right)\left(\frac{T}{n}+\frac{n-2i+1}{2}\tau\right)+(1-p)(T+n\tau) = \\
&\quad = p\left(\frac{T}{n^2}\sum_{i=1}^{n} i+\frac{1}{n}\sum_{i=1}^{n} i\tau\frac{2n-3i+3}{2}+\frac{\tau^2}{4T}\sum_{i=1}^{n} i(n-i+2)(n-2i+1)\right)+(1-p)u(T+n\tau).
\end{aligned}$$

Since

$$\frac{T}{n^2}\sum_{i=1}^{n} i = \frac{Tn(n+1)}{2n^2} = \frac{T(n+1)}{2n}, \tag{68}$$

$$\begin{aligned}
\frac{1}{n}\sum_{i=1}^{n} i\tau\frac{2n-3i+3}{2} &= \frac{\tau}{n}\left(\frac{2n+3}{2}\sum_{i=1}^{n} i-\frac{3}{2}\sum_{i=1}^{n} i^2\right) = \\
&= \frac{\tau}{n}\left(\frac{n(n+1)(2n+3)}{4}-\frac{3}{2}\frac{n(n+1)(2n+1)}{6}\right) = \\
&= \frac{\tau(n+1)}{2}\left(\frac{2n+3}{2}-\frac{2n+1}{2}\right) = \frac{\tau(n+1)}{2},
\end{aligned} \tag{69}$$

and

$$\begin{aligned}
\sum_{i=1}^{n} i(n-i+2)(n-2i+1) &= \sum_{i=1}^{n} i(n^2-3in+3n-5i+2i^2+2) = \\
&= 2\sum_{i=1}^{n} i^3-(3n+5)\sum_{i=1}^{n} i^2+(n^2+3n+2)\sum_{i=1}^{n} i = \\
&= 2\frac{n^2(n+1)^2}{4}-(3n+5)\frac{n(n+1)(2n+1)}{6}+(n+1)(n+2)\frac{n(n+1)}{2} =
\end{aligned}$$

$$= \frac{n+1}{2}(n^2(n+1) - \frac{(3n+5)n(2n+1)}{3} + n(n+1)(n+2)) =$$
$$= \frac{n+1}{6}(3n^2(n+1) - n(3n+5)(2n+1) + 3n(n+1)(n+2)) =$$
$$= \frac{n+1}{6}(3n^3 + 3n^2 - 6n^3 - 13n^2 - 5n + 3n^3 + 9n^2 + 6n) =$$
$$= -\frac{n(n-1)(n+1)}{6}, \tag{70}$$

we can see that

$$\begin{aligned} E(t_1, \ldots, t_n) &= p\left(\frac{n+1}{2n}T + \frac{n+1}{2}\tau - \frac{n(n-1)(n+1)}{24}\frac{\tau^2}{T}\right) + (1-p)(T+n\tau) = \\ &= p\left(\left(\frac{1}{2} + \frac{1}{2n}\right)T + \frac{n+1}{2}\tau - \frac{n^3-n}{24}\frac{\tau^2}{T}\right) + (1-p)(T+n\tau). \end{aligned} \tag{71}$$

Simplifying this equation gives us (34).

Proof of Theorem 10

Let ξ be an extension of n for the real domain and $E(\xi)$ be the corresponding right part of (34). E obtains the optimal (minimal) value when $\frac{dE}{d\xi} = 0$ or when $\xi = 1$ or when $\xi \to \infty$. If $\xi = 1$, then $E = T + \tau$. We will show that $T + \tau$ is not the optimal value for the general case. If $\xi \to \infty$, then from (66) and the formula for E we can see that $E \to \infty$.

We can now calculate the optimal ξ:

$$\frac{dE}{d\xi} = p\left(-\frac{T}{2\xi^2} + \frac{\tau}{2} - \frac{3\xi^2 - 1}{24}\frac{\tau^2}{T}\right) + (1-p)\tau = 0,$$

and therefore

$$-\frac{T}{2\xi^2}p + \left(1 - \frac{p}{2}\right)\tau - \frac{3\xi^2 - 1}{24}\frac{\tau^2}{T}p = 0.$$

Hence

$$\xi^4\frac{\tau^2}{4T} - \xi^2\left(\tau\left(\frac{2}{p} - 1\right) + \frac{\tau^2}{12T}\right) + T = 0,$$

and therefore

$$\begin{aligned} \xi^2 &= \frac{\left(\tau\left(\frac{2}{p}-1\right) + \frac{\tau^2}{12T}\right) \pm \sqrt{\left(\tau\left(\frac{2}{p}-1\right) + \frac{\tau^2}{12T}\right)^2 - \tau^2}}{\frac{\tau^2}{2T}} = \\ &= \frac{1}{6} + \frac{2T}{\tau}\left(\frac{2}{p} - 1\right) \pm \sqrt{\left(2\tau\left(\frac{1}{p} - 1\right) + \frac{\tau^2}{12T}\right)\left(\frac{2\tau}{p} + \frac{\tau^2}{12T}\right)}\frac{2T}{\tau^2} = \\ &= \frac{1}{6} + \frac{2T}{\tau}\left(\frac{2}{p} - 1\right) \pm \sqrt{\left(\frac{8T}{\tau}\left(\frac{1}{p} - 1\right) + \frac{1}{3}\right)\left(\frac{2T}{\tau}\cdot\frac{1}{p} + \frac{1}{12}\right)} = \\ &= \frac{1}{6} + \frac{2T}{\tau}\left(\frac{2}{p} - 1\right) \pm \sqrt{\frac{16T^2}{\tau^2}\left(\frac{1}{p} - 1\right)\frac{1}{p} + \frac{2}{3}\frac{T}{\tau}\left(\frac{2}{p} - 1\right) + \frac{1}{36}}. \end{aligned}$$

So we have

$$\xi_{1,2} = \sqrt{\frac{1}{6} + \frac{2T}{\tau}\left(\frac{2}{p} - 1\right) \pm \sqrt{\frac{16T^2}{\tau^2}\left(\frac{1}{p} - 1\right)\frac{1}{p} + \frac{2}{3}\frac{T}{\tau}\left(\frac{2}{p} - 1\right) + \frac{1}{36}}}.$$

Since we are looking for a local minimum,

$$\frac{d^2E}{d\xi^2} = \frac{T}{\xi^3} - \frac{\xi}{4}\frac{\tau^2}{T} \geq 0,$$

and therefore

$$\xi^4 \leq \frac{4T^2}{\tau^2} \Rightarrow \xi \leq \sqrt{\frac{2T}{\tau}}.$$

It is easy to see that

$$\begin{aligned} \xi_1 &= \sqrt{\frac{1}{6} + \frac{2T}{\tau}\left(\frac{2}{p} - 1\right) + \sqrt{\frac{16T^2}{\tau^2}\left(\frac{1}{p} - 1\right)\frac{1}{p} + \frac{2}{3}\frac{T}{\tau}\left(\frac{2}{p} - 1\right) + \frac{1}{36}}} = \\ &= \sqrt{\frac{1}{6} + \frac{2T}{\tau}} > \sqrt{\frac{2T}{\tau}}, \end{aligned}$$

and therefore only ξ_2 meets this requirement.

We want to prove now that for each value of p and for $T > 0$

$$0 < \xi_2 < \sqrt{\frac{2T}{\tau}}, \tag{72}$$

i.e., the second value is *always* a valid local minimum.

Let us show first that $\xi_2 > 0$. Indeed, this inequality is equivalent to the following one:

$$\frac{1}{6} + \frac{2T}{\tau}\left(\frac{2}{p} - 1\right) > \sqrt{\frac{16T^2}{\tau^2}\left(\frac{1}{p} - 1\right)\frac{1}{p} + \frac{2}{3}\frac{T}{\tau}\left(\frac{2}{p} - 1\right) + \frac{1}{36}}. \tag{73}$$

Taking the square of the both sides of the inequality, we obtain

$$\frac{4T^2}{\tau^2}\left(\frac{4}{p^2} - \frac{4}{p} + 1\right) + \frac{2}{3}\frac{T}{\tau}\left(\frac{2}{p} - 1\right) + \frac{1}{36} > \frac{16T^2}{\tau^2}\left(\frac{1}{p} - 1\right)\frac{1}{p} + \frac{2}{3}\frac{T}{\tau}\left(\frac{2}{p} - 1\right) + \frac{1}{36},$$

and after eliminating equal members we obtain the true inequality

$$\frac{4T^2}{\tau^2} > 0.$$

Therefore, $\xi_2 > 0$. Now we want to prove that

$$\xi_2 < \sqrt{\frac{2T}{\tau}}.$$

As in the previous case, from the formula for ξ_2 we obtain an equivalent inequality

$$\frac{1}{6} + \frac{4T}{\tau}\left(\frac{1}{p} - 1\right) < \sqrt{\frac{16T^2}{\tau^2}\left(\frac{1}{p} - 1\right)\frac{1}{p} + \frac{2}{3}\frac{T}{\tau}\left(\frac{2}{p} - 1\right) + \frac{1}{36}}.$$

Taking the square of both sides of the inequality, we obtain

$$\frac{16T^2}{\tau^2}\left(\frac{1}{p^2}-\frac{2}{p}+1\right)+\frac{4}{3}\frac{T}{\tau}\left(\frac{1}{p}-1\right)+\frac{1}{36}<\frac{16T^2}{\tau^2}\left(\frac{1}{p}-1\right)\frac{1}{p}+\frac{2}{3}\frac{T}{\tau}\left(\frac{2}{p}-1\right)+\frac{1}{36},$$

and after eliminating equal members we obtain the true inequality

$$\frac{16T^2}{\tau^2}\left(1-\frac{1}{p}\right)-\frac{2}{3}\frac{T}{\tau}<0,$$

which is true since $p<1$. Therefore, the presented value ξ_2 represents the single optimal solution of the problem.

Since n must be an integer, we can only use its approximation by either $\lfloor\xi_2\rfloor$ or $\lceil\xi_2\rceil$, with the restrictions by 1 and n_{max}. The function E is smooth enough, so even if both $\lfloor\xi_2\rfloor$ and $\lceil\xi_2\rceil$ are not the real optimal values, they will provide a good enough approximation for E_{opt}.

Proof of Theorem 11

It is easy to see that for the exponential distribution, formula (15) has the form of (40)

$$t_{i+1}=t_i+\frac{e^{\lambda(t_i-t_{i-1})}-1}{\lambda}-\tau \text{ for } i=1,\ldots,n-1.$$

We will now prove by induction the following formula:

$$t_{i+1}=t_i+\frac{1}{\lambda}g_{i-1}(\lambda t_1). \tag{74}$$

For $i=1$ it is trivial. Let us assume the correctness of (74) for $i\leq k-1$. Then for $i=k$ we obtain by (40)

$$\begin{aligned} t_{k+1} &= t_k+\frac{e^{\lambda(t_k-t_{k-1})}-1}{\lambda}-\tau \\ &= t_k+\frac{e^{g_{k-1}(\lambda t_1)}-1}{\lambda}-\tau=t_k+\frac{1}{\lambda}g(g_{k-1}(\lambda t_1)) \\ &= t_k+\frac{1}{\lambda}g_k(\lambda t_1). \end{aligned}$$

It is easy to prove (also by induction) that

$$t_k=t_1+\frac{1}{\lambda}\sum_{i=1}^{k-1}g_i(\lambda t_1). \tag{75}$$

By Lemma 1, $t_n=T$, and formula (41) follows immediately from the previous equation. If we denote this dependency by $v(t_1)=T$, we can see that

$$\frac{dv}{dt}=1+\frac{1}{\lambda}\sum_{i=1}^{n-1}\prod_{j=1}^{i}\lambda e^{g_{j-1}(\lambda t_1)}=1+\sum_{i=1}^{n-1}e^{\sum_{j=0}^{i-1}g_j(\lambda t_1)}>0. \tag{76}$$

Therefore $v(t)$ is a monotonic function, and (41) has a unique solution. By Theorem 4, the solution produced by this point is the global minimum. From equation (76) we can show

by induction that $\frac{dv^p}{dt^p} > 0$ for any p, hence the function is convex down. This fact shows that equation (41) can be solved easily by numerical methods, and the convergence will be fast.

Using (4) we can calculate the value of $E(t)$ at this point:

$$\begin{aligned}
E(t) &= (T+n\tau) - p\sum_{i=1}^{n-1}(t_{i+1}-t_i+\tau)F(t_i) \\
&= (T+n\tau) - p\sum_{i=1}^{n-1}(t_{i+1}-t_i+\tau) + p\sum_{i=1}^{n-1}(t_{i+1}-t_i+\tau)e^{-\lambda t_i} \\
&= (T+n\tau) - p(t_n - t_1 + (n-1)\tau) + p\sum_{i=1}^{n-1}\frac{e^{\lambda(t_i-t_{i-1})}-1}{\lambda}e^{-\lambda t_i} \\
&= (1-p)(T+n\tau) + p\left(t_1+\tau+\frac{1}{\lambda}\sum_{i=1}^{n-1}\left(e^{-\lambda t_{i-1}}-e^{-\lambda t_i}\right)\right) \\
&= (1-p)(T+n\tau) + p\left(t_1+\tau+\frac{1}{\lambda}\left(1-e^{-\lambda t_{n-1}}\right)\right),
\end{aligned}$$

and therefore

$$E(t) = (1-p)(T+n\tau) + p\left((t_1+\tau)+\frac{1}{\lambda}\left(1-e^{-\lambda T+g_{n-1}(\lambda t_1)}\right)\right). \tag{77}$$