Journal of Artificial Intelligence Research 54 (2015) 233–275 Submitted 06/15; published 10/15

Decision Making with Dynamic Uncertain Events

Meir Kalech KALECH@BGU.AC.IL
Department of Information Systems Engineering,
Ben-Gurion University of the Negev, Beer-Sheva, Israel

Shulamit Reches SHULAMIT.RECHES@GMAIL.COM
Department of Applied Mathematics,
Jerusalem College of Technology, Israel

Abstract

When to make a decision is a key question in decision making problems characterized by uncertainty. In this paper we deal with decision making in environments where information arrives dynamically. We address the tradeoff between waiting and stopping strategies. On the one hand, waiting to obtain more information reduces uncertainty, but it comes with a cost. Stopping and making a decision based on an expected utility reduces the cost of waiting, but the decision is based on uncertain information. We propose an optimal algorithm and two approximation algorithms. We prove that one approximation is optimistic - waits at least as long as the optimal algorithm, while the other is pessimistic - stops not later than the optimal algorithm. We evaluate our algorithms theoretically and empirically and show that the quality of the decision in both approximations is near-optimal and much faster than the optimal algorithm. Also, we can conclude from the experiments that the cost function is a key factor to chose the most effective algorithm.

1. Introduction

There are many real-world domains in which an agent has to choose an alternative among multiple candidates, based on their utility. The problem becomes more complicated when the utility depends on events that occur dynamically and therefore the decision itself is based on dynamically changing and uncertain information. In such domains, the question is whether to stop at a particular point and make the best decision given the information available, or wait until more information arrives to enable making a better decision. This problem is not trivial when there is a cost associated with waiting.

For example, consider the problem of finding the best stock to buy in the stock market. The values of the stocks may change over time due to future events, such as publication of the company's sales report or a change in the interest rate, etc. The longer we wait, the more information becomes available and, as a result, a decision can be made with more certainty. In many real-world domains, there is a cost to waiting. For instance, in the above example, the cost of the stock reflects the loss caused by not investing money in one of the candidate stocks. Thus, there is a tradeoff between a waiting strategy that enables one to acquire more information and decreases the uncertainty and a stopping strategy, which reduces the cost.

Another example relates to scheduling systems for meetings. Determining the best time for a meeting could depend on many factors, such as the times of other meetings, location, and the schedule of the attendees. Typically, these factors may change dynamically and influence a decision on the best time for the meeting; obviously, the longer one waits, the more information becomes

©2015 AI Access Foundation. All rights reserved.

available and the probability of choosing the best time for the meeting is higher. However, waiting incurs a cost of the possibility that the chosen time slot might no longer be available. The goal of this paper is to determine the best time to make a decision which maximizes the expected gain and considers the cost of waiting.

This question, of whether to wait to get more information or not, is also raised in the context of real estate investment. There are many unknown factors that may influence the decision of which real estate property to buy. For example, an infrastructure development in the area (like a railway station), raising/reducting municipal taxes in the area, construction of a polluting factory in the area, etc. There is a question whether to choose a real estate property based on the expected gain or to wait and get the information about the next factor but taking the risk that the properties' prices may increase.

The tradeoff between uncertainty and cost is related to the optimal stopping problem (Ferguson, 1989; Peskir & Shiryaev, 2006), the problem of decision making under bounded-resource (Horvitz, 2001, 2013), the problem of decision making with multiple informative but expensive observations (Krause & Guestrin, 2009; Tolpin & Shimoni, 2010), the Max K-Armed Bandit problem (Cicirello & Smith, 2005) and the ranking and selection problem (Powell & Ryzhov, 2012). Our work copes with the challenge of the stopping problem where multiple alternatives are affected by uncertain information that arrives dynamically. The decision whether to stop or wait, in our problem, depends on the utility affected by the result of certain events that will occur in the next time stamps. Since each alternative is affected by different events we should consider the combination of all possible events, which makes our problem hard and different from others.

In this paper, we:

1. Develop a model for representing the arrival of dynamic information and its influence on the utilities of the candidates.

2. Present an optimal exponential algorithm ($OPTIMAL$) that guarantees the best decision tradeoff between certainty and waiting costs.

3. Propose two polynomial approximation algorithms to solve the problem and provide bounds on their error. We prove that both algorithms evaluate the expected utility from stopping optimally. However, one approximation algorithm is optimistic ($OPTIMISTIC$) in the sense that its waiting evaluation is overestimated. The other algorithm is pessimistic ($PESSIMISTIC$), namely its waiting evaluation is underestimated.

4. Empirically evaluate the optimal and the two approximation algorithms and illustrate the advantages of each one of them.

We empirically evaluate the three algorithms by simulating a stock market scenario. We compare the optimal and approximation algorithms to four other baseline algorithms; one algorithm makes the decision at the beginning of the decision process, the second algorithm decides after all the information is obtained, the third algorithm makes the decision at a random time and the fourth one makes the decision after half of the time steps. We examine the algorithms in terms of the quality of the decision (utility) and runtime.

Our empirical evaluation shows that the cost function much influences on the quality of the decision. As the cost function increases more moderately (i.e. a root function), the pessimistic algorithm becomes less effective and the optimistic algorithm improves the quality decision and it is even slightly better than the pessimistic one. However, for cost functions that grow linearly or polynomially with the time the pessimistic approach is much better than the optimistic one, and in most cases there is even no significant difference between the quality of the decision made by the pessimistic approximation algorithm and the optimal algorithm. The quality of both approximations is much better than the baseline algorithms. The runtime of the approximation algorithms is polynomial rather than an exponential runtime of the optimal algorithm.

This work is an extension of our previous work (Kalech & Pfeffer, 2010; Reches, Kalech, & Stern, 2011). In this work we expand both the theoretical and empirical parts of the research. In particular, in the theoretical aspect we find and prove the approximation error of the expected gain and the expected wait of the algorithms, and prove the complexity and other properties of the pessimistic and optimistic algorithms. We greatly expanded the evaluation by presenting the influence of different parameters, such as the cost of waiting, and the distribution on the variables. Furthermore, we empirically show the pessimistic and optimistic behaviour of the algorithms.

The paper is organized as follows. In the next section we present the basic fundamentals of the problem and formally define it. In Section 3 we present the optimal algorithm. The optimistic approach is illustrated in Section 4 and the pessimistic approach in Section 5. An empirical evaluation of the algorithms is provided in Section 6. In Section 7 we discuss related work and our conclusion is presented in Section 8.

2. Model Description

To describe the model clearly we use an example inspired by the stock market. Assume a decision maker wishes to choose one stock to purchase among three stocks (c_1, c_2, c_3). The value of the stocks is influenced by future events, such as the consumer price index (CPI), interest rates, etc. The decision maker cannot evaluate the influence of the future events on the stocks with certainty but only with some degree of probability. Obviously, the sooner the decision is taken, the lesser the loss from not investing the money. On the other hand, the longer the waiting time, the more information that can be gathered by knowing the outcome of the expected events and consequently a decision with a greater degree of certainty can be made.

In our model, each decision will be designated by a *candidate*; throughout the paper we refer to the candidate set $C = \{c_1, c_2, ..., c_m\}$. An agent desires to choose an alternative from the set C. A candidate's utility is influenced by information that arrives dynamically. We represent the dynamic information by random variables. A random variable is an event that occurs at a specific time. The different outcomes of the event may influence the utility of the candidate in different ways. As described extensively later, a candidate may be influenced by multiple events. Let us define formally the *timed variable*.

Definition 1 (timed variable) *A* timed variable *is a pair $\langle X_i, t\rangle$, where X_i is a discrete, finite random variable taking values $x_{i_1}, ..., x_{i_k}$ and $t \in T$ represents its time stamp, where T is a discrete time horizon $T = \{0, ..., h\}$ with horizon h.*

Given a timed variable $\langle X_i, t\rangle$, t is the time stamp in which the random variable X_i is assigned one of its possible values $x_{i_1}, ..., x_{i_k}$.

For example, the timed variables in the stock market are future events that influence the utility of the stocks. $\langle X_1, 1\rangle$ may represent an expected decrease in the percentage of the interest rate at time 1, where time 1 represents one month from now. X_1 takes the discrete values $\{x_{1_1} = -0.1, x_{1_2} =$

$0\}$. Another timed variable, $\langle X_2, 2\rangle$, represents the expected prospectus of a specific company in two months from now, which takes the values $\{x_{2_1} = positive, x_{2_2} = negative\}$. One more timed variable $\langle X_3, 3\rangle$, represents the expected change in the percentage of CPI in three months. X_3 takes the values $\{x_{3_1} = 0.1, x_{3_2} = 0.2\}$. In our example, $\langle X_1, 1\rangle$ and $\langle X_3, 3\rangle$ influence candidate c_1 and $\langle X_2, 2\rangle$ influences candidate c_2. A timed variable may affect two or more candidates.

Definition 2 (assignment) *An* assignment *to the timed variable $\langle X, t\rangle$ is an outcome x_i of X. A* global assignment *at time t, denoted σ^t, is an assignment of values to all timed variables whose time stamp is less than or equal to t.*

For example, the global assignment at time 3 (σ^3) may be $X_1 = 0$, $X_2 = positive$, and $X_3 = 0.1$, i.e., at time 1 (after a month) the interest did not change, at time 2 (after two months) the prospectus of the company was positive, and at time 3 (after three months) the CPI increased by 0.1%.

Each candidate's utility depends on a set of timed variables, where different sets lead to different utilities. We use a tree to represent the effect of the timed variables on the utility.

Definition 3 (candidate tree) *A candidate tree ct_i for candidate c_i is a tree. $n_{j,i}$ is a node in ct_i, where i stands for the index of the candidate and j for its index in the tree. The internal nodes are associated with timed variables. The random variable corresponding to node n is denoted $X(n)$ and its time is denoted $\Gamma(n)$. If $n_{j,i}$ is a descendant of $n_{k,i}$ then $\Gamma(n_{k,i}) < \Gamma(n_{j,i})$. The edges going out of node n represent the possible assignments of $X(n)$. Each edge $X = x$ is labeled by the probability outcome denoted by $p(X = x)$. A leaf n is labeled by its utility $\mathcal{U}(n)$, representing the utility of the candidate affected by the assignments from the root to the leaf. CT represents the set of candidate trees.*

We call the assignments on the path that starts at the root of candidate tree ct_i and ends at node $n_{j,i}$ at time $t = \Gamma(n_{i,j})$ "the local assignment of candidate c_i" denoted by σ^t_i. Note that the candidate tree represents an estimate of the effect of the timed variables on the utility of the candidates. Obviously, this estimate may change over time, due to the addition or removal of times variables or re-estimation of the probabilities or the utilities. In this case the candidate trees should be updated.

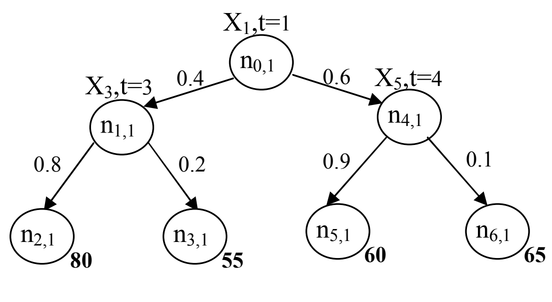


Figure 1: Candidate tree ct_1.

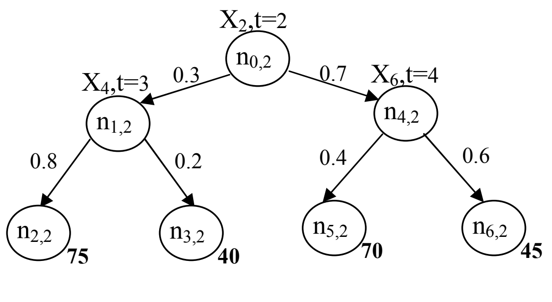


Figure 2: Candidate tree ct_2.

Figures 1 and 2 present two candidate trees built at time stamp $t = 0$. These are an extension of the three timed variables demonstrated above. $n_{0,1}$ is the root of candidate tree ct_1. Its time stamp is $t = 1$, which represents the fact that the random variable X_1 will obtain its outcomes at time 1.

The node $n_{1,1}$ in ct_1 represents the timed variable $\langle X_3, t = 3\rangle$. One possible outcome of the change in the CPI (X_3) is 0.1% (left edge). The probability of this outcome is 0.8. An alternative outcome of the change in the CPI is 0.2% (right edge) and the probability of this outcome is 0.2. The value 80 in the left leaf of ct_1 represents the utility of candidate c_1 when the local assignment at time 3 (σ_1^3) is: the interest decreased by 0.1% ($X_1 = -0.1$) and the CPI increased by 0.1% ($X_3 = 0.1$). The probability of this utility is $0.4 \cdot 0.8 = 0.32$.

The utility of a candidate is known certainly at the leaves. However, the expected utility of a candidate can be calculated beforehand at any time (depth in the candidate tree) and will consider the subtree from that depth. The expected utility can be trivially computed by a recursive function. The expected utility of a leaf is its utility and for an internal node it is the expected utilities of its children. Formally:

Definition 4 (expected utility) *Given a node $n \in ct_i$, the function $\mathcal{EU}(ct_i, n)$ returns the expected utility of n:*

$$\mathcal{EU}(ct_i, n) = \begin{cases} \mathcal{U}(n) & n \textit{ is a leaf} \\ \sum_j p(X(n) = x_j)\mathcal{EU}(ct_i, n_j) & \textit{otherwise} \end{cases}$$

where n_j represents the successor node of n via assignment $X(n) = x_j$.

For instance, the expected utility of the root in Figure 1 is: $\mathcal{EU}(ct_1, n_{0,1}) = 0.4 \cdot (0.8 \cdot 80 + 0.2 \cdot 55) + 0.6 \cdot (0.9 \cdot 60 + 0.1 \cdot 65) = 66.3$.

The expected utility is an estimate of the real utility based on the information known at the current time. Waiting for the next time reduces the uncertainty about the candidates' utilities and hence increases the probability of making a good decision. However, waiting incurs a cost. The cost can be either a function of the assignments or a function of the time. For the sake of simplicity, in this paper we represent the cost as a function of the time. In reality, the cost of waiting for a specific time stamp is usually not higher than the utility gained at that time. Thus, to enforce a realistic cost, we bound the cost by the maximum utility.

Definition 5 (cost) *Given a time stamp $t \in T$, $\mathcal{CST}(t)$ is an increasing function that returns the approximated cost of waiting until time t.*

The expected gain of a node is the difference between the expected utility of the node and the cost of waiting until that node.

Definition 6 (expected gain) *Given node $n \in ct_i$, $\mathcal{GN}(ct_i, n) = \mathcal{EU}(ct_i, n) - \mathcal{CST}(\Gamma(parent(n)))$.*[1] *If n is the root of the candidate tree then: $\mathcal{GN}(ct_i, n) = \mathcal{EU}(ct_i, n)$.*

The reason we reduce the cost of the parent of n, rather than of n is that $\mathcal{EU}(ct_i, n)$ represents the expected gain of the subtree rooted in n without waiting for the outcomes of $X(n)$.

There is a tradeoff between the first component of $\mathcal{GN}$, the expected utility, and the second component, the waiting cost. The objective of this paper is to present an algorithm that will find the

1. Since the utility and cost are not necessarily given in the same scale they should be normalized before reduction. The normalization is domain dependent.

time which maximizes the gain[2]. Unfortunately, we are unable to separate the computation of the optimal time to make the decision and the selection of the best candidate, since the utilities of the candidates depend on future events. Therefore, we define a policy to determine what to do in all situations that the decision maker might face.

Beside the outcomes of the timed variables, the cost function also influences the decision of whether to stop or to wait. A cost function that returns much smaller values than the difference between the expected utilities through time will eventually lead to a 'wait' decision. On the other hand, a cost function that returns values which are too high will eventually lead to a 'stop' decision. For the examples in this paper we use the cost function $\mathcal{CST}(t) = t$. This function ensures proportional cost values (0,1,2,3,4) in relation to the utility values (45–80). For instance, according to the estimate at time stamp $t = 0$, given the global assignment: $X_1 = -0.1$, $X_2 = positive$, the expected gain of candidate c_1 is $\mathcal{GN}(ct_1, n_{1,1}) = 75 - 1 = 74$ and the expected gain of c_2 is $\mathcal{GN}(c_2, n_{1,2}) = 68 - 2 = 66$. As a result, if the decision maker decides to stop it will choose c_1, otherwise it will wait for the next time stamp. The policy dictates the decision on whether to stop or wait.

Definition 7 (policy) *A* policy *is a function $\pi : \sigma \rightarrow \{stop, wait\}$, where σ is the set of all global assignments.*

If the policy specifies to stop, the decision maker chooses the candidate with the current highest expected gain.

Finally, we define the expected gain for the decision maker by using policy π, referred as "global expected gain". To understand this definition we first introduce another definition of the nodes corresponding to a certain time.

Definition 8 ($NODES_t^j$) *The set $NODES_t^j$ represents the following nodes of ct_j: (1) leaves whose parents' time is less than or equal to t: $\{n \in ct_j | \Gamma(parent(n)) \leq t, n \text{ is a leaf}\}$. (2) internal nodes whose parents' time is less or equal to t and their time is greater than t $\{n \in ct_j | \Gamma(parent(n)) \leq t \wedge \Gamma(n) > t, n \text{ is an internal node}\}$.*

For example, in Figure 1 $NODES_3^1 = \{n_{2,1}, n_{3,1}, n_{4,1}\}$, $NODES_4^1 = \{n_{2,1}, n_{3,1}, n_{5,1}, n_{6,1}\}$.

The global expected gain function obtains the candidate trees, the global assignment, and a policy. If the policy specifies to stop, then the global expected gain is the maximum expected gain among the candidates. Otherwise, it recursively computes the expectation of the global expected gain of the different combinations of the roots' children in the next time stamp. Formally:

Definition 9 (global expected gain) *A* global expected gain *is a function that returns the expected gain from choosing policy π:*

$\mathcal{GEG}(CT, \sigma^t, \pi) =$

$$\begin{cases} \max\limits_j \mathcal{GN}(ct_j, n_j) & \text{if } \pi(\sigma^t) = stop \\ \sum\limits_{y \in \{NODES_{t+1}^1 \times, \ldots, \times NODES_{t+1}^m\}} \mathcal{GEG}(CT^y, \sigma^{t_y}, \pi) Pr(y) & \text{if } \pi(\sigma^t) = wait \end{cases}$$

where

2. Although in the decision making theory it is common to maximize the expected utility, here we use the term expected gain, which incorporates the cost in order to distinguish it from the expected utility.

1. *n_j is the root of ct_j.*
2. *CT^y is the set of candidate trees rooted by the nodes in y.*
3. *$Pr(y)$ is the probability of the nodes in y given the assignment σ^t.*
4. *σ^{t_y} represents the union of σ^t and the assignments of $X(n_j)$ which are represented by y.*

In the running example, for the global assignment $\sigma^{t=2}$ and a policy $\pi(\sigma^{t=2}) = stop$ and $y = (n_{1,1}, n_{1,2})$, $\mathcal{GEG}(CT^y, \sigma^t, \pi) = max(73, 66) = 73$. In case of a policy $\pi(\sigma^t) = wait$, we sum $\mathcal{GEG}$ for all possible assignments of the next time. The set of $NODES$ rooted by the nodes $n_{1,1}$ and $n_{1,2}$ at time $t = 3$ are $NODES_3^1 = \{n_{2,1}, n_{3,1}\}$ and $NODES_3^2 = \{n_{2,2}, n_{3,2}\}$, correspondingly, and $y \in \{n_{2,1}, n_{3,1}\} \times \{n_{2,2}, n_{3,2}\}$. Therefore:

$$\begin{aligned}
&\mathcal{GEG}(CT^{\{n_{1,1},n_{1,2}\}}, \sigma^t, \pi) = \\
&\mathcal{GEG}(CT^{\{n_{2,1},n_{2,2}\}}, \sigma^t, \pi) \cdot 0.8 \cdot 0.8 + \\
&\mathcal{GEG}(CT^{\{n_{2,1},n_{3,2}\}}, \sigma^t, \pi) \cdot 0.8 \cdot 0.2 + \\
&\mathcal{GEG}(CT^{\{n_{3,1},n_{2,2}\}}, \sigma^t, \pi) \cdot 0.2 \cdot 0.8 + \\
&\mathcal{GEG}(CT^{\{n_{3,1},n_{3,2}\}}, \sigma^t, \pi) \cdot 0.2 \cdot 0.2
\end{aligned}$$

Based on the above definitions, we can define the timed decision making problem (TDM):

Definition 10 (Timed Decision Making (TDM) problem) *Given a set of candidate trees CT, the TDM problem is to find a policy that maximizes $\mathcal{GEG}(CT, \sigma^0, \pi)$.*

Table 1 summarizes the notation we use in describing the model.

The hardness of the TDM problem is due to the computation of the expected gain from waiting for the next time stamp. This computation needs to take into consideration the utility of each candidate for each possible assignment and for each time stamp. Specifically, at time t_0, the decision maker should decide whether to stop and choose the best candidate or to wait for the next time stamp by comparing the expected gain from stopping and the expected gain from waiting. The expected gain from stopping at time stamp t_0 can be computed immediately by taking the maximum of the expected gain of the candidate trees ($\max_j \mathcal{GN}(ct_j, n_j)$, where $n_1, ..., n_m$ are the roots of the candidate trees). On the other hand, the computation of the expected gain from waiting for the next time stamp is hard. The expected wait considers the combination of the possible assignments of the candidate trees at time t_1. For each such combination, we need to take into consideration the utility of stopping at time t_1 and the utility of waiting for time stamp t_2 which include additional combinations of assignments and so on. The problem is hard because the number of combinations is exponential in the number of candidates. Specifically, at a certain time stamp, for k possible assignments of different variables in m different candidate trees, the number of combinations is k^m.

In order to prove that TDM is NP-hard, we present the timed decision making problem as a decision problem . Given CT, σ^0 and a non-negative integer K. Answer "yes" if there exist a policy π such that the global expected gain $\mathcal{GEG}(CT, \sigma^0, \pi) \geq K$.

Theorem 1: TDM problem is NP-hard. (The proof appears in Appendix A).

One option for representing our problem would be to use Markov Decision Processes (MDP). In such a model, the states at time t would be global assignments at time t (σ^t) and the actions would be either to select the best candidate at that time (stop) or wait one more time step. The transition

Parameter	**Description**
$C = \{c_1, ..., c_m\}$	A set of m candidates.
$\langle X_i, t\rangle$	A timed variable, where X_i is a discrete, finite random variable and t is its time stamp.
ct_i	A candidate tree of c_i.
CT	A set of candidate trees.
σ^t	An assignment of values to all timed variables whose time stamp is less than or equal to t.
σ_i^t	A local assignment. The assignments on the path that stats at the root of candidate ct_i and end at node whose time is t t.
$n_{j,i}$	A node in the candidate tree ct_i of candidate c_i.
$X(n)$	The random variable corresponding to node n.
$\Gamma(n)$	The time of the random variable corresponding to node n.
$T = (0, ..., h)$	The time horizon.
$\mathcal{U}(n_{ij})$	The utility of candidate ct_i affected by the assignments from its root to the leaf n_{ij}.
$\mathcal{EU}(ct_i, n)$	The expected utility of candidate tree ct_i from node n.
$\mathcal{CST}(t)$	The approximated cost of waiting until time t.
$\mathcal{GN}(ct_i, n)$	The expected gain of node n in candidate tree ct_i is the difference between the expected utility of the node and the cost of waiting until that node.
A *policy*	A function $\pi : \sigma \rightarrow \{stop, wait\}$.
$NODES_t^j$	A set of nodes whose parents' time is less than or equal to t and they are leaves, or their time is greater then t.
$\mathcal{GEG}$	Global expected gain.
$\mathcal{EW}(\sigma^t, \pi)$	The expected gain from waiting for the next time stamp (t+1), using policy π.
$\mathcal{ES}(\sigma^t, \pi)$	The expected gain from stopping at time stamp t, using policy π.
$\mathcal{PTH}(ct_i, n)$	A path, set of local assignments from the root of ct_i to node n.
$Pr\mathcal{PTH}(ct_i, n)$	The probability of a path.
π_i	Local policy of candidate c_i.

Table 1: The notation used for the description of the model.

function from a time t state to a time $t+1$ state for a wait action would be given by the product of the probabilities of time $t+1$ assignments. A stop action leads to a terminal state in which a reward is received equal to the gain of the winning candidate.

The usual advantage of an MDP formulation is the possibility of using dynamic programming methods, such as value and policy iteration. However, in our problem, dynamic programming provides no benefits because the same state cannot be reached by different paths and the number of states is exponential in the total number of timed variables in all the trees.

One of the methods which addresses large MDPs is by means of factored MDPs (Boutilier, Dearden, & Goldszmidt, 2000; Guestrin, Koller, Parr, & Venkataraman, 2003). This approach is not viable in our domain because the utility of stopping is a maximization over the utilities in all the trees which depends on all the timed variables. As we show in Section 4, there is a special structure in our problem that is not readily apparent in the MDP or a factored MDP formulation.

3. Optimal Algorithm

The optimal gain can be calculated straightforwardly by a decision tree approach. The optimal decision tree merges the candidate trees into a single decision tree whose depth is the maximal time of the timed variables in the candidate trees. In the decision tree, we define three kinds of nodes:

Decision nodes, in which the decision maker must decide whether to stop or to wait; if the decision is to stop, then which candidate to choose.

Stop nodes, where the decision maker has stopped and chosen one of the candidates.

Wait nodes, where the decision maker has decided to wait.

Each node is marked with a time stamp. Edges leading out of the wait nodes are labeled by conjunctions of assignments. Every node in the tree is marked by a set of assignments, which are the assignments on the path leading to the node.

The tree is constructed at offline time as follows:

Procedure 1 Optimal:

1. *The root is a decision node with time stamp 0.*
2. *For each time stamp, the children of decision nodes with time stamp t are the wait nodes with time stamp t; for each candidate there is a stop node child. If t is the final time stamp, no wait node child is included.*
3. *Stop nodes are leaves of the tree. If the stop node corresponds to node $n \in ct_i$ at time t, the value of the node is $\mathcal{GN}(ct_i, n)$ (Definition 6).*
4. *The children of a wait node with time stamp t given a global assignment σ^t are determined as follows:*
    - *(a) The local assignment σ_i^t passes through a path of assignments ending at a node. Let us denote this node by n_i for candidate tree ct_i.*
    - *(b) Let $X_n = \bigcup_i X(n_i)$.*

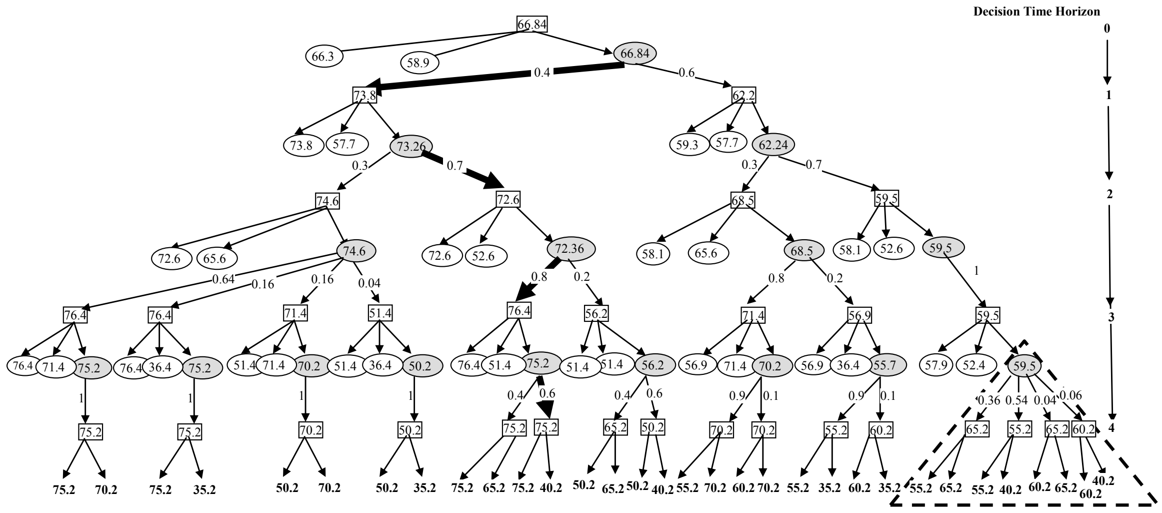


Figure 3: Optimal decision tree for ct_1 and ct_2 candidate trees.

(c) *For each possible* joint *outcome of the timed variables in X_n, the wait node has a child, labeled by the joint probability. The child is a decision node with time stamp $t + 1$.*

Once the tree has been constructed, it can be evaluated using a simple bottom-up process. The gains at the leaves, i.e., the stop nodes, have already been calculated. The gain of a wait node is the expectation of the utilities of its children. The gain of a decision node is the maximum gain of its children and the optimal decision is the one that leads to a maximal gain. The decision tree is generated and evaluated in advance before any assignments have been undertaken. Its solution represents a policy (Definition 7).

Figure 3 presents the optimal decision tree for a decision problem with candidate trees ct_1 (Figure 1) and ct_2 (Figure 2). The time line on the right of the graph represents the time horizon of the decision. The rectangular nodes represent decision nodes; the shaded ellipse nodes represent wait nodes and the empty ellipse nodes represent stop nodes[3]. The stop nodes come in pairs, one for each candidate; the node for c_1 is on the left. The numbers in the nodes represent the expected gains and are computed using the bottom-up algorithm. In the example, we used a cost function that linearly grows in time: $\mathcal{CST}(t) = 1.2\dot{t}$, so the cost of reaching a leaf is 4.8 (since there are four time stamps).

For example, consider the dashed triangle on the right-hand side of the figure. The root of the subtree shown in this triangle is a wait node with time stamp 3. In determining the children of this node, we consider all timed variables in the candidate trees with a time stamp of 4. There are two such timed variables, $\langle X_5, 4\rangle$ for candidate c_1 and $\langle X_6, 4\rangle$ for candidate c_2. We need to split all the joint outcomes of these two candidates so that the wait node has four children. Each of these children is a decision node with time stamp 4. Since this is the last time stamp, these decision nodes only have stop nodes as their children.

One particular course of events is shown in bold in the figure. Since the expected gain of waiting (66.84) is higher than the expected gain from stopping, which is the expected utility from choosing

3. The stop nodes at the final time do not have ellipses for readability. We have also omitted assignment labels on the edges.

the best stop node (66.3), the agent will wait. Accordingly, the assignment $X_1 = 0$ (left) will then occur. At the next decision node child, the expected gain from stopping and choosing candidate c_1 (73.88) is higher than the expected gain of waiting (73.26), so the agent may stop at $t = 1$ and choose c_1. Assume the sequence of assignments occurred as depicted in bold in the figure. This eventually will lead to a leaf node with a gain of 75.2 as shown. Note, however, that this gain incorporates the cost of waiting 4 time stamps. Thus, if the agent had been omniscient and known the outcomes of the timed variables in advance, it would have obtained a gain of 80, since it would not have had to wait at all. For a rational agent which stops at time 1, the gain for choosing c_1 is $80 - 1.2 = 78.8$.

Denote π^* as the optimal policy. Given a global assignment σ^t, we use $\mathcal{EW}(\sigma^t, \pi^*)$ to represent the expected gain from waiting for the next time stamp by executing the optimal algorithm. It equals the wait node given σ^t. Similarly, we use $\mathcal{ES}(\sigma^t, \pi^*)$ to represent the expected gain from stopping at time stamp t by executing the optimal algorithm; this equals the maximal stop node given σ^t. For example, in Figure 3, $\mathcal{EW}(\sigma^0, \pi^*) = 66.84$ and $\mathcal{ES}(\sigma^0, \pi^*) = 66.3$.

The optimal decision tree explicitly represents the state space in an MDP model, as described in Section 2. Since the state space is represented in a tree (rather than a graph), an intelligent value iteration process is equivalent to the backward induction algorithm we use for the optimal decision tree.

3.1 Analysis

The time complexity of the optimal algorithm is affected by the fact that the optimal decision tree considers the different combinations of paths of the candidate trees. Let the maximum size of a candidate tree be M and the number of candidates be m. Notice that the size M of a candidate tree is exponential in the depth of the local candidate tree. Specifically, given the depth of the candidate tree, h (the horizon), and the number of outcomes for each timed variable, k (the branching factor) then $M = k^h$. The total number of timed variables is Mm and the size of the depth of the optimal decision tree is bounded by $m \log(M)$. The worst-case time complexity of computing the optimal tree is $O(M^m)$.

As mentioned above, the backward induction on the optimal decision tree is equivalent to the value iteration for MDPs. Since the state space is represented in a tree, the value iteration simply scans the state space. Thus, the complexity of a value iteration is the size of the state space. The complexity of a state space, as described in Section 2, is the sum of the global assignment alternatives at each time. In the worst case, where every candidate depends on different timed variables at each time, there are k^m alternatives at time 1, $(k^2)^m$ alternatives at time 2, and $(k^h)^m$ alternatives at time h. This complexity is identical to the complexity of optimal decision tree $O(M^m)$ (since $M \in O(k^h)$).

The best-case complexity is archived when all candidates are affected by the same timed variable, since we should not consider the combinations between the same timed variables. In this case the trees of the candidates are identical except for the utilities in the leaves and the complexity is thus mM.

Beyond the exponential complexity of the optimal algorithm, another disadvantage of this algorithm stems from the fact that every change in the candidate trees demands rebuilding the decision tree. Unfortunately, due to its exponential complexity rebuilding is not feasible. To cope with the exponential complexity of the optimal algorithm and the fact that it is not feasible to rebuild the decision tree when timed variables change, we propose two approximation algorithms in the following sections.

Beyond the exponential complexity of the optimal algorithm, another disadvantage of this algorithm stems from the fact that every change in the candidate trees demands rebuilding the decision tree. The optimal algorithm computes the whole combinations of the future events in advance and as a result, it makes an optimal solution as long as the initial evaluation about the probabilities and utilities of each event is valid. Since the complexity of the optimal algorithm is exponential, it may be infeasible to rebuild a new decision tree for each time stamp. However, in realistic scenarios the evaluation about the events and the utilities may change over time. To cope with the exponential complexity of the optimal algorithm and the fact that it is not feasible to rebuild the decision tree when timed variables change, in the following sections we propose two approximation polynomial algorithms.

4. An Optimistic Approach

The optimal algorithm presented in Section 3 considers all candidates simultaneously and thus grows exponentially in the number of candidates. In this section, we present an alternative algorithmic framework that considers the candidates separately and dynamically. This alternative viewpoint will lead to a more efficient approximation algorithm.

4.1 OPTIMISTIC Algorithm

The main idea behind the alternative framework is calculating the utility of each candidate tree separately and then combining the utilities together to obtain an evaluation of the global gain. In this way we avoid the complexity of comparing each assignment to all the assignments of the other candidates; each candidate contributes separately to the overall utility. Specifically, a candidate contributes to the overall utility only when it actually prevails over all the other candidates. Thus we can estimate the utility from a node in a candidate tree by the product of its expected gain and the probability that the candidate will win, given that the node is reached. Then, in order to estimate the overall gain, we sum over this utility for all the candidates. To formally describe the algorithm we present the following definitions:

Definition 11 (path) *Given a node $n \in ct_i$, the function $\mathcal{PTH}(ct_i, n)$ returns a set of local assignments from the root to n $\{X_{i_1} = x_{i_1}, X_{i_2} = x_{i_2}, ...\}$ in candidate tree ct_i.*

Definition 12 (probability of path) *Given a node $n \in ct_i$, $Pr\mathcal{PTH}(ct_i, n) = \prod_{j \in \mathcal{PTH}(ct_i, n)} Pr(j)$.*

The probability that c_i will prevail over a specific candidate c_j at time t is the sum of its probabilities to prevail over c_j for each possible assignment, i.e., for each node in $NODES_t^j$. The probability that a candidate c_i will prevail over the other candidates, given a specific node $n_{x,i} \in ct_i$ and at a specific time stamp, $t = \Gamma(n_{x,i})$, is the sum of the probabilities that it will prevail over each candidate in the current time t. Formally[4] :

4. For the mathematical calculations of the probabilities in the approximation algorithms, we assume that the candidates have disjointed sets of timed variables that are probabilistically independent; which means that two candidates are not affected by the same time variable. Nevertheless, as will be shown by the results of the experiments, the algorithms perform well even when there are common variables.

Definition 13 (probability of winning) *Given a node $n_{x,i} \in ct_i$, the probability of c_i to win is*[5]*:*

$$Pr(c_i \textit{ wins}|n_{x,i}) = \prod_{j\in\{1,\dots,m\},i\neq j} \sum_{n_{y,j}\in NODES_t^j} IsWin(\mathcal{EU}(ct_i, n_{x,i}), \mathcal{EU}(ct_j, n_{y,j}))Pr\mathcal{PTH}(ct_j, n_{y,j})$$

where

1. $t = \Gamma(parent(n_{x,i}))$.
2. $IsWin(\mathcal{EU}(ct_i, n_{x,i}), \mathcal{EU}(ct_j, n_{y,j})) =$

$$\begin{cases} 1 & \textit{if } \mathcal{EU}(ct_i, n_{x,i}) > \mathcal{EU}(ct_j, n_{y,j}) \\ 0 & \textit{else} \end{cases}$$

For example, using Figures 1 and 2 above, recall $NODES_3^1 = \{n_{2,1}, n_{3,1}, n_{4,1}\}$.

$$\begin{array}{l} Pr(c_2 \text{ wins}|n_{2,2}) = \\ IsWin(\mathcal{EU}(ct_2, n_{2,2}), \mathcal{EU}(ct_1, n_{2,1})) \cdot 0.4 \cdot 0.8 + \\ IsWin(\mathcal{EU}(ct_2, n_{2,2}), \mathcal{EU}(ct_1, n_{3,1})) \cdot 0.4 \cdot 0.2 + \\ IsWin(\mathcal{EU}(ct_2, n_{2,2}), \mathcal{EU}(ct_1, n_{4,1})) \cdot 0.6 = \\ 0 \cdot 0.32 + 1 \cdot 0.08 + 1 \cdot 0.6 = 0.68 \end{array}$$

We define the "relative expected gain" of each candidate as its contribution to the global expected gain given a specific node.

Definition 14 (relative expected gain) *Given a node $n_{x,i} \in ct_i$, the relative expected gain of candidate c_i is $\mathcal{GN}(ct_i, n_{x,i}) \cdot Pr(c_i \textit{ wins}|n_{x,i})$.*

Notice that the probability of candidate c_i to win at a time stamp t is:

$$Pr(c_i \text{ wins}) = \sum_{n_{x,i}\in NODES_t^i} Pr(c_i \text{ wins}|n_{x,i}) \cdot Pr(n_{x,i})$$

and thus according to the law of total probability:

$$\sum_i \sum_{n_{x,i}\in NODES_t^i} Pr(c_i \text{ wins}|n_{x,i}) \cdot Pr(n_{x,i}) = 1$$

The computation of the relative expected gain of $n_{x,i}$ is presented in Algorithm 1. In line 3 we go over the candidate trees except for candidate tree ct_i. In line 5 we go over the candidate tree's nodes that have the time as the same as $n_{x,i}$'s time. We then sum over the probabilities of those nodes whose expected utility is less than that of $n_{x,i}$ (lines 6–8). This sum represents the probability of c_i to win c_j, given the node $n_{x,i}$. Finally, in line 10 we multiply the probability that c_i will prevail over all the candidates (given the node $n_{x,i}$), since a winning candidate should prevail over all the candidates. We return the product of this probability and the expected gain of the node

5. Ties between candidates are broken in a consistent manner.

Algorithm 1 RELATIVE_EXPECTED_GAIN
(**input**: candidate trees $CT = \{ct_1, ..., ct_m\}$
input: node $n_{x,i}$
output: relative expected gain of $n_{x,i}$)

```
 1: $t \leftarrow \Gamma(n_{x,i})$
 2: $prob \leftarrow 1$
 3: for all $ct_j \in CT$ $(i \neq j)$ do
 4:     $temp \leftarrow 0$
 5:     for all $n_{y,j} \in NODES_t^j$ do
 6:         if $\mathcal{EU}(ct_i, n_{x,i}) > \mathcal{EU}(ct_j, n_{y,j})$ then
 7:             $temp \leftarrow temp + Pr\mathcal{PTH}(ct_j, n_{y,j})$
 8:         end if
 9:     end for
10:     $prob \leftarrow prob \cdot temp$
11: end for
12: return $prob \cdot \mathcal{GN}(ct_i, n_{x,i})$
```

$n_{x,i}$. For instance, the relative expected gain from choosing candidate c_2 given the node $n_{2,2}$ in time 3 (with a cost function $\mathcal{CST}(t) = 1.2\dot{t}$) is $(75 - 3.6) \cdot 0.68 = 48.552$.

The optimistic approach determines the policy (Definition 7) by constructing a separate decision tree for each candidate. The policy then is determined based on the local assignment for each time. As mentioned earlier the local assignment for a certain candidate tree is derived from the global assignment. The estimate of the global expected gain from stopping and from waiting depends on the policy and the expected gain of each candidate separately. We call such a policy "local policy". Obviously, it may be possible that at a certain time the local policies over the candidates will be different.

Definition 15 (local policy) *A* local policyπ_i *for candidate c_i is a rule that dictates either stopping or waiting for a local assignment σ_i^t.*

For each assignment at time t, the optimistic decision maker decides on its policy by building individual decision trees based on the relative expected gain for each candidate. The expected gain from stopping at each time is the sum of the relative expected gain of the candidates. The optimistic procedure is invoked first with time 0:

Procedure 2 Optimistic:

1. *Generate an individual candidate decision tree for each candidate c_i based on ct_i in a manner similar to the optimal decision tree except that the* **relative expected gain** *is used instead of the* **expected gain**.

2. *Denote the stop node of the current time stamp t for each candidate tree ct_i by $\mathcal{ES}^i(\sigma^t, \pi_i)$ and the wait node of the current time stamp t for each candidate tree ct_i by $\mathcal{EW}^i(\sigma^t, \pi_i)$. Denote also:*
   $\mathcal{ES}(\sigma^t, \pi) = \sum_{i \in \{1,...,m\}} \mathcal{ES}^i(\sigma^t, \pi_i)$ *and*
   $\mathcal{EW}(\sigma^t, \pi) = \sum_{i \in \{1,...,m\}} \mathcal{EW}^i(\sigma^t, \pi_i)$.

3. *If* $\mathcal{ES}(\sigma^t, \pi) \geq \mathcal{EW}(\sigma^t, \pi)$
   then $\pi = stop$, *return* $\underset{i \in \{1,\ldots,m\}}{\operatorname{argmax}} \mathcal{ES}^i(\sigma^t, \pi_i)$
   else $\pi = wait$.

4. *Prune each candidate tree according to the global assignment such that the tree will be rooted by the node reached by the local assignment. Invoke Procedure 2 in the next time stamp* $t + 1$.

If we consider the stop values at the root, exactly one candidate will have a probability of winning of 1 and all others will have a probability of 0. Therefore, the expected gain of the winning candidate equals the sum of the values of the stop nodes in the root of all the decision trees. Intuitively, the value of a wait node for a candidate is an estimate of the candidate's contribution to the benefit of waiting. Therefore, the algorithm evaluates the expected utility from waiting by summing up the expected wait of the candidates. If the summation value is greater than the maximum immediate expected gain, then the total expected gain of waiting is greater than the expected gain of stopping. In this case the decision will be to wait. Otherwise, the decision will be to stop and to choose the candidate with the highest expected gain.

If the agent decides to wait then the decision trees must be updated according to the new assignments obtained after waiting. The new assignments prune some parts of the candidate trees. For instance, consider candidate tree ct_1 in Figure 1. Assume that the agent decided to wait for the outcome of the variables at time 1. Assume the assignment of the timed variable X_1 at time 1 is the left, then the right subtree of ct_1 can be pruned since it no longer influences the utility of ct_1. The $NODES^1_t$ set, which is associated with a specific time, changes as a result of this pruning as well as the computation of the relative expected gain. Therefore rebuilding the decision trees is necessary.

Figures 4 and 5 present the decision tree of ct_1 and ct_2, respectively (Figures 1 and 2). The leaves contain two numbers; the first represents the expected utility while the second (in bold) represents the relative expected gain.

For example, let us compute the relative expected gain of the rightmost bottom-level node $n_{6,1}$ in Figure 1. The cost function is $\mathcal{CST}(t) = 1.2 \cdot t$. This node, with a utility of 65, is greater than the two nodes in ct_2 $n_{3,2}$ with a utility of 40 and $n_{6,2}$ with a utility of 45. The total probability of the two nodes in ct_2 to be defeated by $n_{6,1}$ is $0.3 \cdot 0.2 + 0.7 \cdot 0.6 = 0.48$. This is the probability that c_1 will defeat c_2 given its utility of 65. To compute the relative expected gain of this node (see Definition 14), we multiply the gain, which is $65 - 4.8 = 60.2$, by the probability of winning, resulting in 28.9 (see the rightmost bottom-level node in Figure 4).

Based on the decision trees in Figures 4 and 5 we find that the Optimistic algorithm decision at time 0 is to wait, since the sum of the wait nodes in $t = 0$ (89.3) is greater than the sum of the stop nodes (66). Suppose that timed variables $\langle X_1, 1\rangle$ have been assigned by the left outcome.(as in the example of the optimal algorithm). The decision trees are now updated. Candidate c_1's tree is pruned so that it only includes the subtree rooted at $n_{1,1}$ and as a result the relatives expected gain of candidate trees c_1 and c_2 being updated. Figures 6 and 7 present the obtained trees by the Optimistic algorithm at this case. According to this trees, the algorithm at time stamp $t = 0$, decides again to wait, since the sum of the wait nodes (84), is higher than the sum of the stop nodes(73.8). At the next decision node child, the expected gain from stopping and choosing candidate c_1 (73.88) is higher than the expected gain of waiting (73.2), so the agent may stop at $t = 1$ and choose c_1.

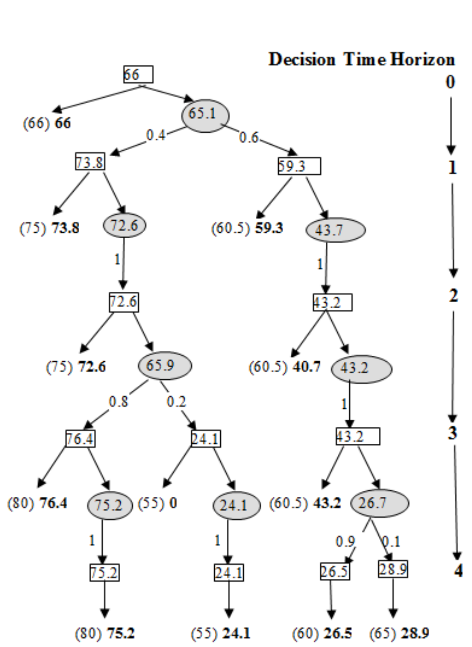


Figure 4: Candidate decision tree for ct_1 (built at $t = 0$).

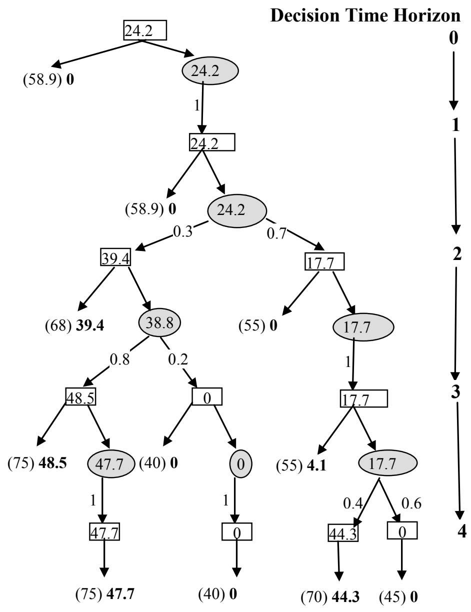


Figure 5: Candidate decision tree for ct_2 (built at $t = 0$).

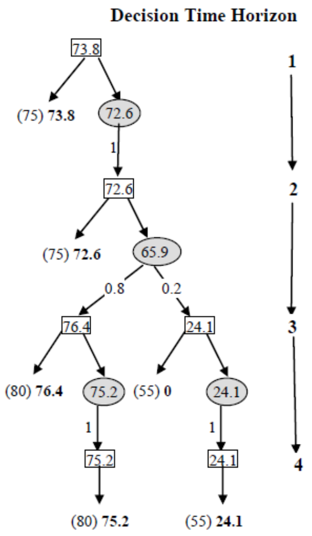


Figure 6: Candidate decision tree for ct_1 (rebuilt at $t = 1$).

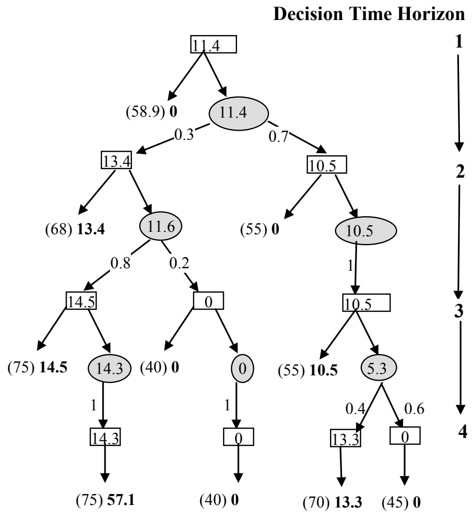


Figure 7: Candidate decision tree for ct_2 (rebuilt at $t = 1$).

4.2 Analysis

The time complexity of the optimistic approximation is only polynomial in the number of candidates since we build a decision tree for every candidate separately.

Theorem 1 *The time complexity of building the decision trees in the optimistic approximation is $O(M^2m^2)$, where m is the number of candidates and M is the maximal size among the candidate trees.*

Proof: When evaluating each candidate tree, we must compute the probability of winning at $O(M)$ nodes. For each such node, we perform a summation over $O(M)$ nodes in each of the other candidate trees and its cost $O(M^2m)$. We should perform this for all m candidate trees. Thus, the total cost of the algorithm is $O(M^2m^2)$. $\square$

This result compares favorably to $O(M^m)$ for the optimal algorithm if the number of candidates is large. In addition, due to its polynomial complexity and due to the fact that the optimistic algorithm rebuilds the decision trees to update the probabilities, it can easily update the decision tree by new dynamic events or by updated probabilities and utilities. For instance, assume that at time $t = 0$ the prediction is that the interest rate will increase at time $t = 2$ in 0.1% with a probability of 0.8 and in 0.2% with a probability of 0.2. At time $t = 1$ this prediction may change, for instance, to 0.15% with a probability of 0.6 and 0.2% with a probability of 0.4. Since the optimistic algorithm rebuilds in polynomial time the decision trees it can easily consider the updated probabilities and values.

We will show now that if Procedure 2 returns a stopping policy, an optimal algorithm would decide the same. When the algorithm waits, it implies that the expected gain from waiting is greater than the expected gain from stopping. In this case, the optimal waiting expectation could be lower.

Theorem 2 *Given a global assignment σ^t, if π is a policy obtained by Procedure 2 and π^* is an optimal policy, then $\mathcal{ES}(\sigma^t, \pi) = \mathcal{ES}(\sigma^t, \pi^*)$ and $\mathcal{EW}(\sigma^t, \pi) \geq \mathcal{EW}(\sigma^t, \pi^*)$.*

Proof: First we prove that $\mathcal{ES}(\sigma^t, \pi) = \mathcal{ES}(\sigma^t, \pi^*)$. Procedure 2 calculates $\mathcal{ES}(\sigma^t, \pi)$ at time stamp t by summing the stop nodes $n_{x,i}$ of all the candidate trees ct_i at this time: $\mathcal{ES}(\sigma^t, \pi) = \sum_{i \in \{1,\ldots,m\}} \mathcal{ES}^i(\sigma^t, \pi_i)$. These nodes are the relative expected gain from stopping at this time and, according to definition 14, are $\mathcal{GN}(ct_i, n_{x,i})Pr(c_i \text{ wins}|n_{x,i})$. Since the global assignment of this time is known, exactly one candidate c_i (the candidate with the highest expected gain at time t) confirms $Pr(c_i \text{ wins}|n_{x,i}) = 1$ and all others confirm $Pr(c_j \text{ wins}|n_{x,j} \in ct_j) = 0$. As a result, $\mathcal{ES}(\sigma^t, \pi) = \mathcal{GN}(ct_i, n_{x,i})$, where c_i is the candidate with the highest expected gain at time t. Thus, $\mathcal{ES}(\sigma^t, \pi) = \mathcal{ES}(\sigma^t, \pi^*)$.

We now prove that $\mathcal{EW}(\sigma^t, \pi) \geq \mathcal{EW}(\sigma^t, \pi^*)$. The optimistic approach estimates the expected waiting $\mathcal{EW}(\sigma^t, \pi)$ by the sum $\sum_i \mathcal{EW}^i(\sigma^t, \pi_i)$. Since for every decision tree of ct_i and for any global assignment σ^t the optimistic approach chooses the policy π_i that maximizes $\mathcal{EW}^i(\sigma^t, \pi_i)$ (looks for the optimal time to stop for each local assignment and takes the combination of the utilities) independently of the other candidate trees, there is a possibility that the sum $\sum_i \mathcal{EW}^i(\sigma^t, \pi_i)$ includes a relative expected gain of one candidate from stopping at a specific time stamp i and a relative expected gain of another candidate from waiting till time stamp $i + 1$ for the same global assignment. Since each relative expected gain is optimal, then $\mathcal{EW}(\sigma^t, \pi) \geq \mathcal{EW}(\sigma^t, \pi^*)$.$\square$

Corollary 1 *Based on the last theorem, given a policy obtained by Procedure 2, if $\pi(\sigma^t) = stop$, an optimal policy would decide the same. This is because when an optimistic policy π decides to stop, $\mathcal{ES}(\sigma^t, \pi) > \mathcal{EW}(\sigma^t, \pi)$. Then, based on the last theorem, $\mathcal{EW}(\sigma^t, \pi) \geq \mathcal{EW}(\sigma^t, \pi^*)$ and $\mathcal{ES}(\sigma^t, \pi) = \mathcal{ES}(\sigma^t, \pi^*)$, thus $\mathcal{ES}(\sigma^t, \pi^*) > \mathcal{EW}(\sigma^t, \pi^*)$, namely an optimal policy will declare a* *stopping policy too. Therefore, the optimistic approach guarantees the optimal expected gain from stopping.*

We now prove the approximation error of the expected wait. Notice that the following theorem does not discuss the error of the optimistic algorithm but focuses on the worst case error when estimating the waiting gain by the optimistic algorithm.

Theorem 3 *Given a time horizon $T = (0, ..., h)$, if π is a policy obtained by Procedure 2 and π^* is a policy obtained by the optimal algorithm, $\mathcal{EW}(\sigma^0, \pi) - \mathcal{EW}(\sigma^0, \pi^*) \leq \sum_{i=1}^{f-1} \mathcal{EU}(ct_i, n_i) + \mathcal{CST}(h)$, where n_i is the root of candidate tree ct_i, $\mathcal{EU}(ct_i, n_i)$ is the expected utility of the node n_{ij} in candidate tree ct_i and $f = Min(m, h)$ (m is the number of candidates).*

Proof: According to the optimistic approach, $\mathcal{EW}(\sigma^t, \pi) = \sum_i \mathcal{EW}^i(\sigma^t, \pi_i)$. Since the expected wait of each candidate $\mathcal{EW}^i(\sigma^t, \pi_i)$ is computed independently, the global expected wait $\mathcal{EW}(\sigma^t, \pi)$ may include, for a specific assignment, the stopping gain of one candidate and the waiting gain of another candidate simultaneously (even though this combination is impossible).

The worst case scenario, whereby $\mathcal{EW}(\sigma^t, \pi)$ has the highest value occurs when for each time stamp, exactly one local policy π_i is $\pi_i(\sigma^t) = stop$. In this situation, the expected wait is the sum of the expected stop in different time stamps of f candidates, where $f = Min(m, h)$. Thus, (1) $\mathcal{EW}(\sigma^t, \pi) \leq \sum_{i=1}^{f} \mathcal{EU}(ct_i, n_i)$, where n_i is the root of candidate tree ct_i. Now, $\mathcal{EW}(\sigma^t, \pi^*) \geq \mathcal{EU}(ct_j, n_j) - \mathcal{CST}(h)$ when $\mathcal{EU}(ct_j, n_j)$ is the highest expected utility among the roots of the candidate trees. Thus, (2) $-\mathcal{EW}(\sigma^t, \pi^*) \leq -\mathcal{EU}(ct_i, n_j) + \mathcal{CST}(h)$ and as a result, by summing (1) and (2), $\mathcal{EW}(\sigma^t, \pi) - \mathcal{EW}(\sigma^t, \pi^*) \leq \sum_{i=1}^{f-1} \mathcal{EU}(ct_i, n_i) + \mathcal{CST}(h)$. In particular, for $t = 0$: $\mathcal{EW}(\sigma^0, \pi) - \mathcal{EW}(\sigma^0, \pi^*) \leq \sum_{i=1}^{f-1} \mathcal{EU}(ct_i, n_i) + \mathcal{CST}(h)$. □

Finally, we prove the approximation error of the global expected gain. It is actually the cost of waiting till level l_{h-1}, when l_h is the last level.

Theorem 4 *Given time horizon $T = (0, ..., h)$, a policy π obtained by Procedure 2,a cost function $\mathcal{CST}(t)$, a set of candidate trees CT, a global assignment σ^t and an optimal policy π^*, the global expected gain $\mathcal{GEG}$ holds: $\mathcal{GEG}(CT, \sigma^0, \pi^*) - \mathcal{GEG}(CT, \sigma^0, \pi) \leq \mathcal{CST}(h-1)$.*

Proof: According to Corollary 1, Procedure 2 guarantees an optimal policy for $\pi(\sigma^t) = stop$. As a result, an error can be obtained only when $\pi(\sigma^t) = wait$ and $\pi^*(\sigma^t) = stop$. Since waiting for the next time stamp decreases uncertainty, the error is only the cost of waiting. In the worst case scenario, Procedure 2 may wait until the last time stamp while an optimal policy would stop immediately. However, the policy obtained by Procedure 2 at time $h-1$ is optimal, $\mathcal{GEG}(CT, \sigma^{h-1}, \pi) = \mathcal{GEG}(CT, \sigma^{h-1}, \pi^*)$. The reason behind this is that because the value of $\mathcal{EW}(\sigma^{h-1}, \pi)$ considers only local policies $\pi_i(\sigma^{h-1}) = wait$, the estimated expected wait from this policy, which is the sum of the local expected wait, is optimal, since it does not include the relative gain of waiting and stopping for the same assignment.

As a result, the absolute approximation error is $\mathcal{GEG}(CT, \sigma^0, \pi^*) - \mathcal{GEG}(CTs, \sigma^0, \pi) < \mathcal{CST}(h-1)$.□

5. A Pessimistic Approach

In this section we present an alternative approximation algorithm which, in contrast to the former, presents a pessimistic approach. This algorithm considers the expected utility of each time stamp separately. As a result, we avoid an exponential complexity of the optimal algorithm which considers all the possible combinations between waiting and stopping for each time stamp.

5.1 PESSIMISTIC Algorithm

The approximation gain can be calculated by a united decision tree. In this approach we merge candidate utility functions into a single decision tree, where each level in the tree represents a time stamp associated with a timed variable, i.e., level l_i in the tree represents a time point t_i where we decide whether to stop or wait. In the decision tree there are two nodes on each level:

Stop node, where the decision maker stops and chooses one of the candidates. A stop node, $\mathcal{ES}_i$, is the expected utility of stopping at level l_i.

Wait node, where the decision maker decides to wait. The wait node, $\mathcal{EW}_i$, is the expected utility of waiting for the next time level. This is the maximum between the stop node and the wait node of level l_{i+1}.

In our approximate solution for each time stamp we compute the expected utility of stopping ($\mathcal{ES}_i$) at that level. When stopping, the optimal choice is the candidate with the highest expected utility. To compute the expected utility of stopping ($\mathcal{ES}_i$) optimally, we should compute the expected utility of the winning candidate in each possible assignment and multiply it by the probability of that assignment. A brute force approach will consider all the combinations between the assignments of the timed variables and for each one return the product of the winner's expected utility and the probability of the assignment. This approach is obviously exponential in the number of candidates since the size of the assignment combinations is exponentially affected by the number of candidates.

Alternatively, we can relax the time complexity by sorting the expected utilities of each candidate. In this way we can easily find the winner and multiply its expected utility by the probabilities of the assignments of the other candidates with a lower expected utility. Since the expected utilities are sorted, this computation is linear in the number of candidates. In Theorem 5 we analyze the time complexity in detail.

To describe this calculation in Algorithm 2, we use the definition of $NODES$ (see Definition 8). The algorithm obtains time t and a set of candidate trees CT and returns the expected utility of stopping at that time. For each one of the candidate trees, lines 8–11 sort the nodes with times less or equal to t ($NODES_t^j$) according to their expected utility. The sorting is done in an inverse order and the ordered nodes are inserted into array $s_j[]$. All the arrays are added to set S. In order to iterate over the arrays, we initiate pointers to the arrays; $indx[]$ contains m pointers to m arrays, where $indx[j]$ contains a pointer to array $s_j[]$. All the pointers are initiated to point at the first node in the corresponding array (lines 12–14). In the main loop (lines 15–19), we find the node with the highest expected utility among the nodes that are pointed at. To compute its probability of winning, we multiply its own probability with the lower probabilities of the nodes of the other candidates. Namely, for each one of the candidate trees, we sum over the probabilities of the nodes that have a lower expected utility than the winner (line 17) and multiply this summation with the probability of the node that currently wins. The sum of the probabilities in the array of candidate c_i ($s_i[]$) that are lower than that of c_j is actually the probability that c_j is greater than c_i and thus its probability to beat it. Since the arrays are sorted, this summation is actually done from the current pointer to the end of the arrays. This follows the law of total probability. Finally, in line 18, we increment the

Algorithm 2 EXPECTED_STOPPING

```
(input: time $t$)
(input: candidate trees $CT = \{ct_1, ..., ct_m\}$)
output: expected stopping $\mathcal{ES}(\sigma^t, \pi)$
 1: Internal variables:
 2: $S \leftarrow \emptyset$
 3: $indx[m]$
 4: $i \leftarrow 1$
 5: $j \leftarrow 1$
 6: $exp \leftarrow 0$
 7: $best$
 8: for all $ct_j \in CT$ do
 9:     $s_j[] \leftarrow$ sort $NODES_t^j$ in inverse order
10:     $S \leftarrow S \bigcup s_j[]$
11: end for
12: for all $j \leq m$ do
13:     $indx[j] \leftarrow 1$
14: end for
15: while $\forall j \leq m,\ indx[j]$ did not reach the end of $s_j[]$ do
16:     $best \leftarrow k$, where k confirms$s_k[indx[k]] \geq s_i[indx[i]]\ \forall i \in \{1, ..., m\}$
17:     $exp \leftarrow exp + \mathcal{EU}(ct_{best}, s_{best}[indx[best]]) \cdot Pr\mathcal{PTH}(ct_{best}, s_{best}[indx[best]]) \cdot \prod_{i \neq best} \sum_{k=indx[i]}^{|s_i[]|} Pr\mathcal{PTH}(ct_i, s_i[k])$
18:     $indx[best] \leftarrow indx[best] + 1$
19: end while
20: return $exp - \mathcal{CST}(t)$
```

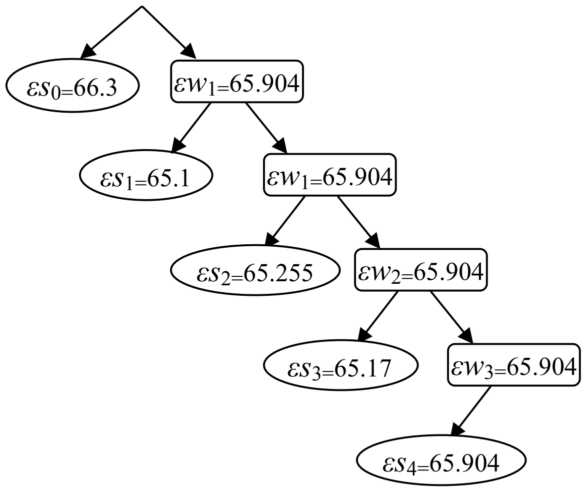


Figure 8: Pessimistic approach: decision tree based on ct_1 and ct_2.

pointer of the local winner to the next node to find the winner in the next iteration. We continue this loop until one of the candidate nodes has been scanned. In this case, all the unscanned utilities of the other arrays are less than the last utility in the array that has been completely scanned. In line 20 the function subtracts the cost of waiting from exp. This algorithm will be demonstrated in the net page.

The next procedure describes the pessimistic decision tree for time stamp t. Such a tree is rebuilt for each time stamp. It is invoked first with time 0:

Procedure 3 Pessimistic:

1. *Generate a decision tree in a bottom-up manner:$\mathcal{ES}_i$ for $i \in \{0, ..., h\}$ is computed based on Algorithm 2. Then, $\mathcal{EW}_{h-1}$ is equal to $\mathcal{ES}_h$ and $\mathcal{EW}_i$ for $i \in \{0, ..., h-2\}$ is the maximum between $\mathcal{ES}_{i+1}$ and $\mathcal{EW}_{i+1}$. This building iterates up to the root at time stamp t. Denote $\mathcal{ES}_t$ and $\mathcal{EW}_t$ by $\mathcal{ES}(\sigma^t, \pi)$ and $\mathcal{EW}(\sigma^t, \pi)$ respectively.*

2. *If $\mathcal{ES}(\sigma^t, \pi) \geq \mathcal{EW}(\sigma^t, \pi)$*
   then, $\pi = stop$, return $\underset{i \in \{1,...,m\}}{\operatorname{argmax}} \mathcal{ES}^i(\sigma^t, \pi_i)$
   else $\pi = wait$.

3. *Prune each candidate tree according to the global assignment such that the tree will be rooted by the node reached by the local assignment and invoke Procedure 3 with time stamp $t + 1$.*

Let us demonstrate the approximate decision tree (Figure 8). Figure 1 and Figure 2 represent two candidate trees and $\mathcal{CST}(t) = 1.2 \cdot t$. We generate the decision tree in a bottom-up manner since each waiting node is actually the maximum of the nodes in the next level. In the last level l_4, there is only one node, $\mathcal{ES}_4$. In order to calculate the expected utility of stopping, we use Algorithm 2. $NODES_4^1 = \{n_{2,1}, n_{3,1}, n_{5,1}, n_{6,1}\}$, $NODES_4^2 = \{n_{2,2}, n_{3,2}, n_{5,2}, n_{6,2}\}$. Algorithm 2 sorts these sets into s_1 and s_2: $s_1 = [80, 65, 60, 55]$, $s_2 = [75, 70, 45, 40]$. In the first iteration (lines 15–19), the pointer to the winner is $best = 1$ since $s_1[1] = 80 > s_2[1] = 75$. Thus,

$$\begin{aligned} exp = \quad & s_1[1] \cdot Pr\mathcal{PTH}(ct_1, n_{2,1}) \cdot \\ & (Pr\mathcal{PTH}(ct_2, n_{2,2}) + Pr\mathcal{PTH}(ct_2, n_{3,2}) + \\ & Pr\mathcal{PTH}(ct_2, n_{5,2}) + Pr\mathcal{PTH}(ct_2, n_{6,2})) = \\ & 80 \cdot (0.4 \cdot 0.8) \cdot (0.3 \cdot 0.8 + 0.7 \cdot 0.4 + 0.7 \cdot 0.6 + 0.3 \cdot 0.2) = 25.6 \end{aligned}$$

Then the pointer of s_1 is incremented to point to $s_1[2]$. In the next iteration, $best = 2$ since $s_2[1] = 75 > s_1[2] = 65$. Thus,

$$\begin{aligned} exp = \quad & exp + s_2[1] \cdot Pr\mathcal{PTH}(ct_2, n_{2,2}) \cdot \\ & (Pr\mathcal{PTH}(ct_1, n_{6,1}) + Pr\mathcal{PTH}(ct_1, n_{5,1}) + Pr\mathcal{PTH}(ct_1, n_{3,1})) = \\ & exp + 75 \cdot (0.8 \cdot 0.3) \cdot (0.6 \cdot 0.1 + 0.6 \cdot 0.9 + 0.4 \cdot 0.2) = \\ & exp + 12.24 = 37.84 \end{aligned}$$

Lastly, $exp = 70.704$ and the expected utility of stopping is $\mathcal{ES}_4 = 70.704 - 4.8 = 65.904$. $\mathcal{EW}_3 = \mathcal{ES}_4$, since there is no wait node in time t_4. Similarly, according to Algorithm 2, we calculate $\mathcal{ES}_3$ based on $NODES_3^1 = \{n_{2,1}, n_{3,1}, n_{4,1}\}$, $NODES_3^2 = \{n_{2,2}, n_{3,2}, n_{4,2}\}$: $\mathcal{ES}_3 = 65.172$. $\mathcal{EW}_2 = max(\mathcal{ES}_3, \mathcal{EW}_3) = 65.904$. The complete decision tree is presented in Figure 8.

At runtime, the decision maker decides to wait or stop according to $\mathcal{ES}_0$ and $\mathcal{EW}_0$ (in the first iteration when $t = 0$). The agent decides to stop if $\mathcal{ES}_0 > \mathcal{EW}_0$ and then it chooses the candidate with the highest expected utility. If the agent decides to wait, several assignments will occur. At this point the decision tree needs to be recomputed as several nodes have become irrelevant. In the presented example the agent decide to stop at time stamp $t = 0$ since the expected stop, $\mathcal{ES}_0$ (66.3) is higher than the expected wait $\mathcal{EW}_0$ (65.904).

5.2 Analysis

First, we show the time complexity of the pessimistic approach.

Theorem 5 *Given time horizon $T = (0, ..., h)$, the time complexity of building the decision tree in the pessimistic approximation is $O(h \cdot (m^2M + mM \log M))$, where m is the number of candidates and M is the maximal size among the candidate trees.*

Proof: At time stamp t_i, the algorithm sorts the nodes in the set $NODES^j_{t_i}$ for each candidate tree ct_j. Since the maximum number of nodes in $NODES^j_{t_i}$ is M, the worst case complexity of this sort is $M \log M$. Since we perform this sort for each candidate tree, the complexity is $O(mM \log M)$. To compare sorted sets in set S, the algorithm goes over the candidates and finds the maximum among the pointed nodes of the candidates. This computation is m^2. The algorithm stops once it reaches the end of one candidate's array (line 15). The worst case is M. Finding $Pr\mathcal{PTH}$ of each node can be calculated once before the loop with a complexity of $mM \log M$. Thus, the worst case time complexity of Algorithm 2 is $O(m^2M + mM \log M)$. We perform Algorithm 2 for each time stamp and as a result the time complexity is $O(h \cdot (m^2M + mM \log M))$.$\square$

Similar to the optimistic algorithm, the pessimistic algorithm rebuilds the decision tree in each time stamp in polynomial time and thus can address changed and additional timed variables.

We now show that if Procedure 3 decides to wait an optimal algorithm would operate similarly. If the expected gain from stopping is greater than the expected gain from waiting, Procedure 3 returns a stopping policy. In this case an optimal policy could return a waiting policy.

Theorem 6 *Given a cost function $\mathcal{CST}(t)$, a set of candidate trees CT, a global assignment σ^t and an optimal policy π^*, a policy π taken by Procedure 3 and an optimal policy π^*, the global expected gain $\mathcal{GEG}$ holds: $\mathcal{ES}(\sigma^t, \pi) = \mathcal{ES}(\sigma^t, \pi^*)$ and $\mathcal{EW}(\sigma^t, \pi) \leq \mathcal{EW}(\sigma^t, \pi^*)$.*

Proof: For each time t, Algorithm 2 calculates the expected gain from stopping, $\mathcal{ES}(\sigma^t, \pi)$, by summing for each possible assignment the expected utility of the candidate with the highest value (the winner candidate) times the probability of the assignment. Since for each time stamp the sum of the probabilities of all the possible assignments is 1, according to the law of total probability (Beaver & Mendenhall, 1983), $\mathcal{ES}(\sigma^t, \pi) = \mathcal{ES}(\sigma^t, \pi^*)$. The waiting node is the maximum between the wait node and the stop node at the next time level. Therefore, the algorithm does not take into consideration the combination of waiting and stopping for different assignments. In contrast, with an optimal policy the algorithm considers such combinations and takes the maximal value for each assignment. Thus, the value of its expected wait may be higher and as a result the wait node's value is less than or equal to the optimal expected gain from waiting, $\mathcal{EW}(\sigma^t, \pi) \leq \mathcal{EW}(\sigma^t, \pi^*)$.$\square$

Corollary 2 *Based on the last theorem, if $\pi(\sigma^t) = wait$, an optimal policy would result in the same decision. This is due to the fact that if π is a policy obtained by Procedure 3 and $\pi(\sigma^t) = wait$, then $\mathcal{EW}(\sigma^t, \pi) > \mathcal{ES}(\sigma^t, \pi)$. Based on the last theorem $\mathcal{EW}(\sigma^t, \pi) \leq \mathcal{EW}(\sigma^t, \pi^*)$ and $\mathcal{ES}(\sigma^t, \pi) = \mathcal{ES}(\sigma^t, \pi^*)$, and thus $\mathcal{EW}(\sigma^t, \pi^*) > \mathcal{ES}(\sigma^t, \pi^*)$ therefore, policy π^* decides to wait. Consequently, the pessimistic approach guarantees the optimal expected gain from waiting.*

We now prove the approximation error of the expected wait.

Theorem 7 *Given a time horizon $T = (0, ..., h)$, a cost function $\mathcal{CST}(t)$, a set of candidate trees CT, a global assignment σ^t and a global assignment σ^t, if π is a policy obtained by Procedure 3 and π^* is a policy obtained by the optimal algorithm, $\mathcal{EW}(\sigma^t, \pi^*) - \mathcal{EW}(\sigma^t, \pi) \leq \mathcal{CST}(h) - \mathcal{CST}(t + 1)$.*

Proof: The optimal expected gain that can be obtained from stopping or waiting at a specific time stamp t is the expected gain from waiting until the last time stamp h (where there is no uncertainty) without considering the cost of this waiting, $\mathcal{ES}(\sigma^h, \pi) + [\mathcal{CST}(h) - \mathcal{CST}(t)]$. Thus, the expected gain from waiting at time t holds, (1) $\mathcal{EW}(\sigma^t, \pi^*) \leq \mathcal{ES}(\sigma^h, \pi) + [\mathcal{CST}(h) - \mathcal{CST}(t+1)]$ (since the expected wait $\mathcal{EW}(\sigma^t, \pi^*)$ already includes the cost of waiting from time t to time stamp $t+1$). On the other hand, because the wait nodes are calculated as the maximum among their children and since in the last level l_h there is only a stop node, $\mathcal{ES}(\sigma^h, \pi) \leq \mathcal{EW}(\sigma^t, \pi)$ and thus, (2) $-\mathcal{EW}(\sigma^t, \pi) \leq -\mathcal{ES}(\sigma^h, \pi)$. As a result, (by summing the two inequalities (1) and (2)), $\mathcal{EW}(\sigma^t, \pi^*) - \mathcal{EW}(\sigma^t, \pi) \leq \mathcal{CST}(h) - \mathcal{CST}(t+1)$. □

Finally, we prove the approximation error of the global expected gain.

Theorem 8 *Given time horizon $T = (0, ..., h)$, a policy π taken by Procedure 3 and an optimal policy π^*, a cost function $\mathcal{CST}(t)$, a set of candidate trees CT, a global assignment σ^t holds: $\mathcal{GEG}(CT, \sigma^0, \pi^*) - \mathcal{GEG}(CT, \sigma^0, \pi) < \mathcal{CST}(h) - \mathcal{CST}(1)$.*

Proof: According to Corollary 2 , the pessimistic policy π is not optimal except for assignment σ^t, $\pi(\sigma^t) = stop$ and an optimal policy π^* holds $\pi^*(\sigma^t) = wait$. In this case, $\mathcal{GEG}(CT, \sigma^t, \pi^*) = \mathcal{EW}(\sigma^t, \pi^*)$ and $\mathcal{GEG}(CT, \sigma^t, \pi) = \mathcal{ES}(\sigma^t, \pi)$. If $\pi(\sigma^t) = stop$ at time stamp t, then $\mathcal{ES}(\sigma^t, \pi) > \mathcal{EW}(\sigma^t, \pi)$ and according to Theorem 7, $\mathcal{EW}(\sigma^t, \pi^*) - \mathcal{EW}(\sigma^t, \pi) \leq \mathcal{CST}(h) - \mathcal{CST}(t+1)$. Thus, $\mathcal{EW}(\sigma^t, \pi^*) - \mathcal{ES}(\sigma^t, \pi) \leq \mathcal{CST}(h) - \mathcal{CST}(t+1)$. As a result, $\mathcal{GEG}(CT, \sigma^t, \pi^*) - \mathcal{GEG}(CT, \sigma^t, \pi) < \mathcal{CST}(h) - \mathcal{CST}(t+1)$. In particular, for t=0, $\mathcal{GEG}(CT, \sigma^0, \pi^*) - \mathcal{GEG}(CTs, \sigma^0, \pi) < \mathcal{CST}(h) - \mathcal{CST}(1)$.□

6. Evaluation

Before presenting an empirical evaluation, we summarize the theoretical analysis of our algorithms in Table 2.

Policy	**#trees**	**Complexity m-#candidates, M-size of the candidate tree**	**Approximation error of $\mathcal{GEG}$ (h-max time horizon)**	**Expected wait**	**Approximation error of $\mathcal{EW}$**
OPTIMAL	1	$O(M^m)$	0	optimal	0
OPTIMISTIC	m	$O(M^2 m^2)$	$\mathcal{CST}(h-1)$	overestimate	$\sum_{i=1}^{f} \mathcal{EU}(ct_i, n_i) + \mathcal{CST}(h)$
PESSIMISTIC	1	$O(m^2 M + mM \log M)$	$\mathcal{CST}(h) - \mathcal{CST}(1)$	underestimate	$\mathcal{CST}(h) - \mathcal{CST}(t+1)$

Table 2: Summary of the theoretical evaluation of the algorithms.

The three algorithms, OPTIMAL, OPTIMISTIC, and PESSIMISTIC, are based on a decision tree approach. However, while the optimal and the pessimistic algorithms use a single decision tree that merges all the candidates, the optimistic algorithm implements m decision trees, one for each candidate tree. The time complexity and the approximation error of the global expected gain are presented in columns three and four, respectively. The fifth column presents an evaluation of the expected wait. Obviously, the optimal algorithm computes the expected wait optimally. The optimistic algorithm overestimates the expected wait and thus a waiting decision is not optimal since the real expected wait may be less than the stop. The pessimistic algorithm underestimates the expected wait and although a waiting decision is optimal, a stopping decision is not since the real expected wait may be higher and consequently the optimal decision would be to wait. The last column presents the approximation error of the expected wait.

As shown in table 2, the error of the expected wait estimated by the optimistic algorithm is much higher than that of the pessimistic algorithm. As a result we estimate that the pessimistic performances will be closer to the optimal algorithm in most of the situations. However, since the expected wait of the optimistic algorithm is overestimated, it may frequently choose to wait and obtain information and thus in case that the cost function increases moderately in time, the performance of the optimistic algorithm will increase.

6.1 Experimental Settings

We experimentally validated our algorithm within a systematic artificial framework inspired by the stock market. We varied the number of candidate stocks (2–30) and the time horizon of the economic events (1–5) (i.e., the timed variables). We ran each combination 25 times. In each test, the possible profits from the stocks (the utility) were randomly selected from a uniform distribution over the range $[\$10K \ldots \$100K]$. Later we present experiments with additional distributions. We ran each scenario (of 25 tests) with 25 random assignments for the timed variables. Each data point in the graphs is an average of 625 tests (25 random utilities $\times$ 25 random assignments).

We compared the three algorithms (OPTIMAL, OPTIMISTIC and PESSIMISTIC) to four baseline algorithms:

1. A trivial stopping strategy; determining the winning candidate at the beginning based only on the expected utility (STOP).
2. A trivial waiting strategy; determining the winning candidate at the end based on full information (WAIT).
3. An algorithm that stops in the middle ($horizon/2$) and chooses the best candidate based on the expected gain (MIDDLE).
4. An algorithm that stops at a random time (RANDOM).

We compared the above algorithms using two metrics: (1) runtime, and (2) the outcome utility. The runtime of OPTIMAL is the runtime of building the decision tree; the runtime of the approximations is the average runtime of building the decision trees in each level. To normalize the utility, we divided it by the utility gained by an omniscient decision maker with no cost. All the following experiments presented in the next sections apart from those presented in Section 6.4 deal with disjoint timed variables, namely, two candidate trees do not share the same timed variable. Notice that, using disjoint time variables, is the worst case scenario for OPTIMAL since finding the optimal time to stop requires taking into consideration all the combinations between the timed variables of the candidate trees. When the timed variables are disjoint the number of comparisons is exponential in the number of candidates (see Table 2).

6.2 The Effect of the Cost

The cost function is a key factor in selecting the most affective algorithm. To examine this factor, we set a simple cost function that grows linearly with the time $\mathcal{CST}(t) = a \cdot t$ and varied the coefficient of the time stamp (a) from $0.01K$ to $2.91K$, with jumps of $0.15K$. We fixed both the number of candidates and the horizon at 5. STOP strategy, as presented in Figure 9, is not affected by the cost since it stops at time $t = 0$ in any case. On the other hand, the utility of WAIT, MIDDLE and

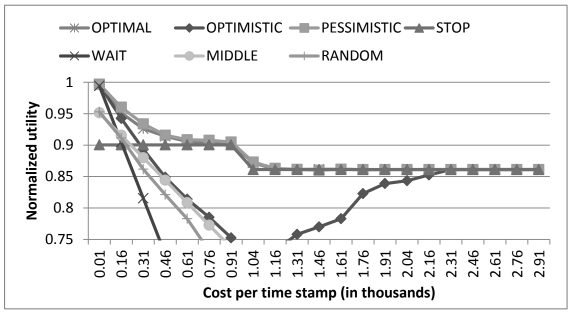


Figure 9: Normalized utility over the cost of each time step, where $\mathcal{CST}(t) = x \cdot K \cdot t$. Time horizon is 5 levels and the number of candidates is 5.

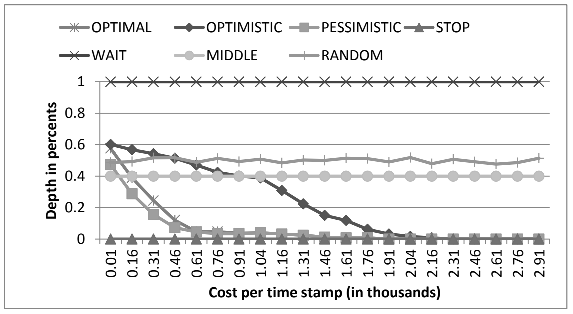


Figure 10: Depth of decision over the cost of each time step, where $\mathcal{CST}(t) = x \cdot K \cdot t$. Time horizon is 5 levels and the number of candidates is 5.

RANDOM, linearly decreases while the cost increases. Figure 10 shows that OPTIMAL and both approximations make the decision earlier as the cost increases since it becomes less worthwhile to wait. However, the depth of the decision decreases faster in OPTIMAL and PESSIMISTIC than in OPTIMISTIC. The depth of the decision influences the utility of the algorithms. It is interesting to see that the gap between the utility of OPTIMISTIC and PESSIMISTIC grows as the cost increases, similarly to the gap in the depth. This can be explained by the fact that OPTIMISTIC overestimates the expected wait and thus it makes the decision later and the loss from the cost is more significant. Nevertheless, from $cost = 2$ both approximations and the optimal algorithm stop at time stamp 0 and achieve the same utility.

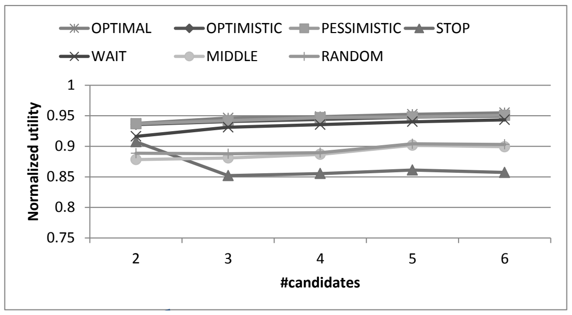


Figure 11: Normalized utility over the number of candidates, where $\mathcal{CST}(t) = 0.28K \cdot \sqrt[3]{t}$ for time horizon of 5 levels.

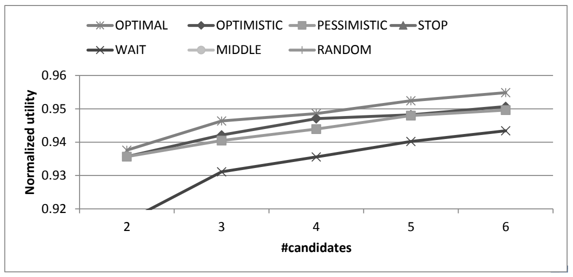


Figure 12: Normalized utility over the number of candidates, where $\mathcal{CST}(t) = 0.28K \cdot \sqrt[3]{t}$ for time horizon of 5 levels: zoom in the utility range of 0.92–0.96.

We further ran experiments with additional cost functions. Figures 11 and 13 present non-linear cost functions. The cost in Figure 11 increases moderately ($\mathcal{CST}(t) = 0.28K \cdot \sqrt[3]{t}$) and the cost in Figure 13 increases fast ($\mathcal{CST}(t) = 0.28K \cdot t^2$). We show that, for the cost function that

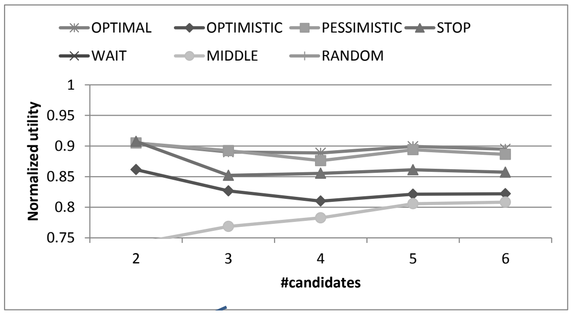


Figure 13: Normalized utility over the number of candidates, where $\mathcal{CST}(t) = 0.28K \cdot t^2$ for time horizon of 5 levels.

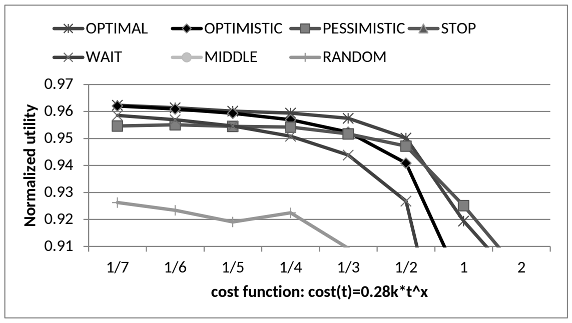


Figure 14: Normalized utility over the cost function: $cost(t) = 0.28K \cdot t^x$, where time horizon is 5 levels and number of candidates is 5.

increases more moderately (a root function), the pessimistic algorithm becomes less effective and OPTIMISTIC becomes better than PESSIMISTIC. As shown in Figure 11 and in a zoom-in view in Figure 12, in such functions OPTIMAL is significantly better than PESSIMISTIC and in most situations it is better than OPTIMISTIC (tested with a 95% confidence value).

To further examine the influence of the cost function on the algorithms, we varied consistently the cost function. We choose the cost function $cost(t) = 0.28K \cdot t^x$ while changing x in the range of $\{\frac{1}{7}, \frac{1}{6}, \frac{1}{5}, \frac{1}{4}, \frac{1}{3}, \frac{1}{2}, 1, 2\}$. Obviously, by decreasing x the function increases more moderately. Figure 14 presents the results. It is clear shown that OPTIMISTIC is better than PESSIMISTIC for root functions smaller than square root. Then, as the cost function increases faster the gap between OPTIMISTIC and PESSIMISTIC increases in a favor of PESSIMISTIC.

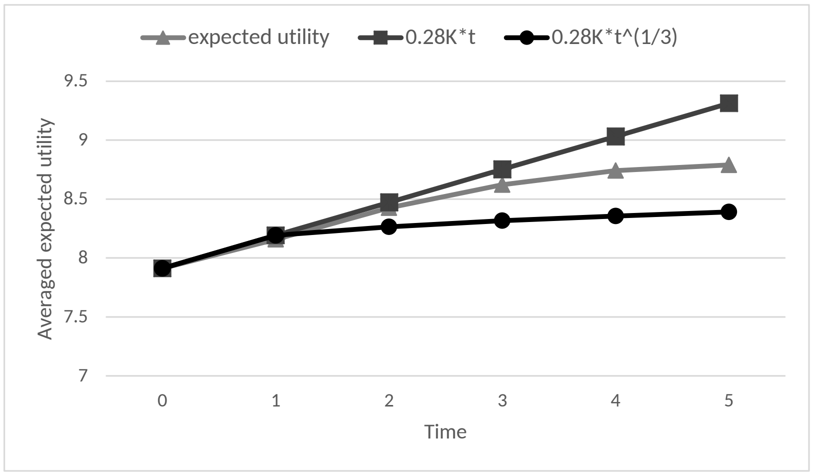


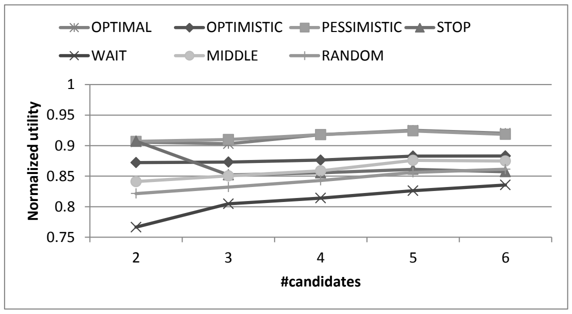
Figure 15: Averaged expected utility over time, where time horizon is 5 levels and number of candidates is 5.

To examine the reason for the behavior of the approximations as dependent in the cost function, we observed the growth of the averaged expected utility as a function of the time (Figure 15). The x-axis is the time horizon and the y-axis is the averaged expected utility. Obviously, the expected utility increase as the time increases since the more events are discovered the uncertainty decreases. It seems that the averaged expected utility grows moderately, approximately logarithmically, as a function of the time. In addition to the averaged expected utility, we present in Figure 15 two cost functions. The first is the linear cost function $\mathcal{CST}(t) = 0.28K \cdot t$, where the pessimistic approximation is better, and the second is the root cost function $\mathcal{CST}(t) = 0.28K \cdot \sqrt[3]{t}$, where the optimistic approximation is better. We set both cost functions to start at the same value of the expected utility on the y-axis. Figure 15 compares between the growth behavior of the two cost functions and the expected utility. This comparison may explain the fact that when the cost function grows linearly the pessimistic algorithm, which usually stops earlier, is better than the optimistic algorithm, since the cost function grows faster than the utility function. However, when the cost function grows more moderately (a root function), meaning, the cost function and the utility function has a similar trend, the optimistic algorithm, which usually stops later, becomes better.

In the rest of the experiments we will examine other factors that influence the performance of the algorithms. As we showed, the difference between the algorithms for a root cost function is small and thus it might be hard to examine the impact of the other factors. Therefore, in the rest of the experiments we use a linear cost function by fixing the waiting cost of all the events to a constant value of $\$2.8K$ for each time stamp ($\mathcal{CST}(t) = 2.8K \cdot t$).

6.3 The Effect of Number of Candidates

We present a subset of the results with a time horizon of 5 levels. Figure 16 presents the utility for a test setting of up to six candidates. Due to memory limitations, the optimal algorithm failed to deal with larger candidate sets. The utility gained by PESSIMISTIC is very close to OPTIMAL and the difference between them is not significant. This result is much better than the results of the baseline algorithms and even better than OPTIMISTIC.



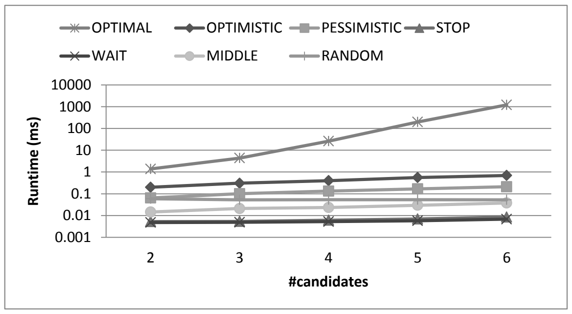


Figure 16: Normalized utility over 6 candidates with a time horizon of 5 levels.

Figure 17: Runtime over 6 candidates with a time horizon of 5 levels.

The runtime is presented in a logarithmic scale in Figure 17. The runtime of all the algorithms is polynomial, except for OPTIMAL, which is exponential. For instance, the average runtime of OPTIMAL for six candidates is 5836 milliseconds, while that of the other algorithms is less than two milliseconds.

We further compared the algorithms, excluding the optimal algorithm, for larger sets of up to 30 candidates. The utility of PESSIMISTIC is always significantly better than the others, as shown in Figure 18. This may be explained by the approximation error of the expected wait. By comparing the approximation error of the two algorithms (see Table 2), it is clear that the approximation error of the expected wait of OPTIMISTIC is much greater than that of PESSIMISTIC and thus OPTIMISTIC is expected to make its decision later than PESSIMISTIC. Note that there is no statistically significant difference between OPTIMISTIC and MIDDLE. Later we will present experiments with larger cost values and horizon where OPTIMISTIC is much better than MIDDLE.

Although the complexity of both PESSIMISTIC and OPTIMISTIC is polynomial, PESSIMISTIC is better than OPTIMISTIC in terms of runtime, as shown in Figure 19. This can be justified by the complexity analysis of the algorithms, as shown in Table 2. While OPTIMISTIC is square in both M and m, PESSIMISTIC is square only in m but not in M.

To illustrate the significance of these results, consider for instance a stock market with five candidate stocks. Based on our experiments, the average utility of the optimal algorithm is $\$92.8K$, which is 97.6% of the utility obtained by an omniscient decision maker. PESSIMISTIC's utility is, on average, less than the optimal by an amount of only $\$300$, while OPTIMISTIC reduces the utility by an amount of $\$2,800$. Obviously, the baseline algorithms reduce the utility drastically. The wait strategy, for instance, produces a profit of only $\$80.5K$.

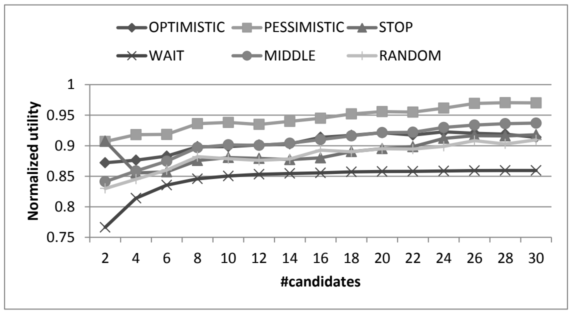


Figure 18: Normalized utility over 30 candidates with a time horizon of 5 levels.

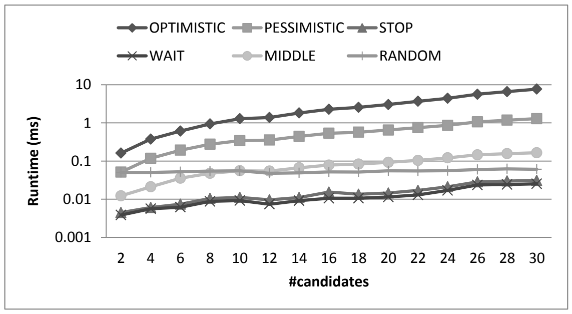


Figure 19: Runtime over 30 candidates with a time horizon of 5 levels.

6.4 Shared Timed Variables

The previous experiments were run on settings where different candidates are not affected by the same timed variable. In the next experiment we show that, even if different candidates are affected by the same timed variables, both approximations achieve a similar utility as previously. We generated candidate trees with a horizon of 5 time stamps. We set 50% of the variables to affect multiple candidates. In Figures 20 and 21 we present the normalized utility for 6 to 30 candidates. By comparing these results to the results without shared variables (Figures 16 and 18) we can see that the baseline algorithms MIDDLE and RANDOM significantly improve their utility. There is no statistically significant difference between each one of the other algorithms with and without shared variables (tested with a 95% confidence value). The reason for the improvement of MIDDLE and RANDOM is that the more shared variables the less uncertainty and thus the expected utilities of the candidates is more accurate. On the other hand the computation of the expected stopping of the

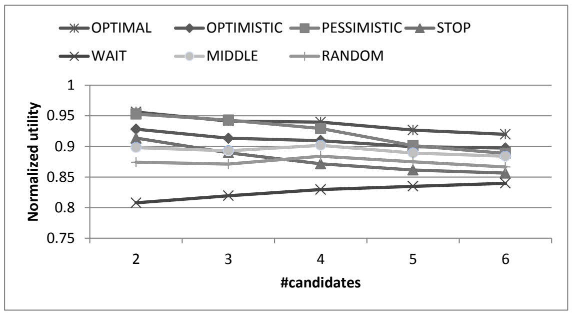


Figure 20: 50% Shared timed variables: Normalized utility over 6 candidates with a time horizon of 5 levels.

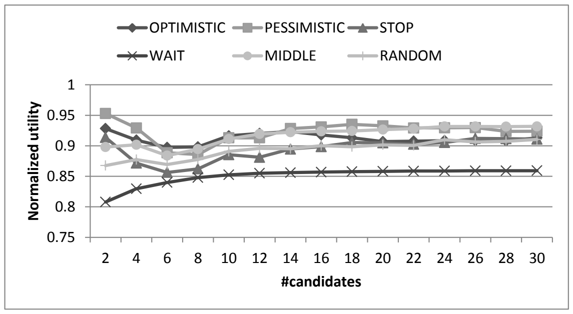


Figure 21: 50% Shared timed variables: Normalized utility over 30 candidates with a time horizon of 5 levels.

approximation algorithms relies on the independence between the variables and thus the decrease in the uncertainty does not improve their results.

We also checked the difference between the algorithms and found that there is no statistically significant difference between OPTIMAL and PESSIMISTIC. Both algorithms are better than OPTIMISTIC but there is no statistically significant difference between OPTIMISTIC and MIDDLE. The results of the approximation algorithms are significantly better than the other baseline algorithms (tested with a 95% confidence value).

As analyzed in Section 3, the optimal algorithm addresses shared timed variables very efficiently and in fact it reduces its computational complexity. In Figure 22 we show the runtime with 50% shared timed variables. Compared to Figure 17 we can see that OPTIMAL runs in two orders of a magnitude faster than experiments with no shared timed variables.

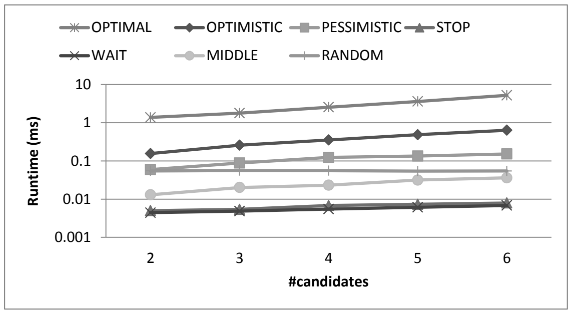


Figure 22: 50% Shared timed variables: runtime over 6 candidates with a time horizon of 5 levels.

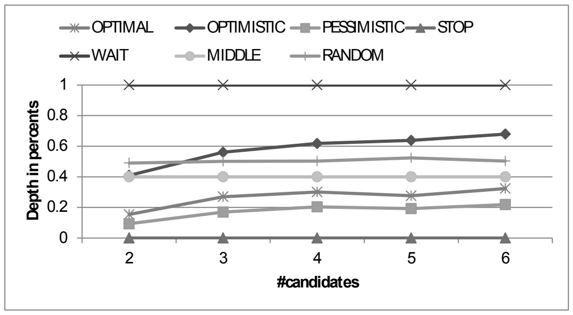


Figure 23: Normalized depth over 6 candidates with a time horizon of 5 levels.

6.5 The Depth of the Decision

Figure 23 illustrates the attributes of the PESSIMISTIC and OPTIMISTIC algorithms when run on candidate trees with a time horizon of 5 levels. The y-axis represents the depth (in percentage relative to the maximal horizon) in which the algorithms stop and decide. Figure 23 presents the results for candidate trees with a time horizon of 5 levels. As analyzed, PESSIMISTIC always stops before the optimal algorithm, since its expected wait is underestimated. OPTIMISTIC always stops after the optimal algorithm since by overestimating the expected wait it continues to wait, although an optimal algorithm decides to stop.

6.6 The Effect of the Horizon

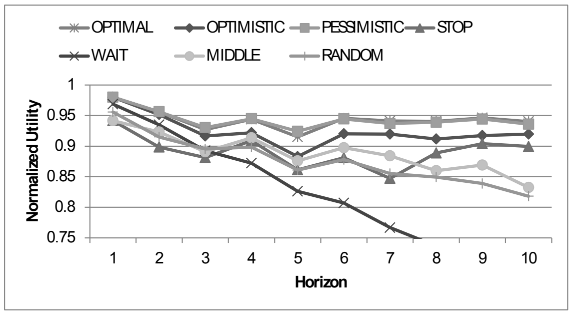


Figure 24: Normalized utility over horizon (1-10) where number of candidates is 5.

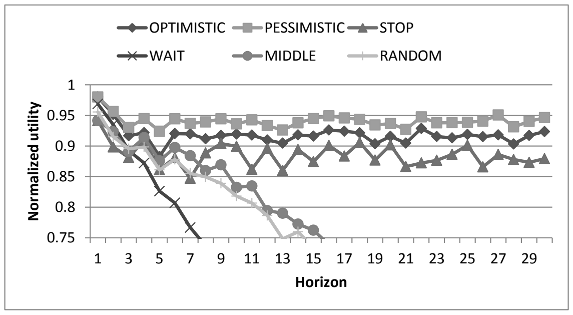


Figure 25: Normalized utility over horizon (1-30) where number of candidates is 5.

Next we examine the influence of the horizon of the candidate trees on the utility. To grow candidate trees with a large horizon, we generated a chain; a tree such that every left branch leads to a leaf while every right branch leads to another internal node. Thus, the size of the candidate tree grows linearly with the horizon. The experimental setting for these experiments includes 5 candidates. Figures 24, 26, and 28 present the results for horizon 1–10 by comparing all the algorithms. Since it was not feasible to run OPTIMAL for a larger horizon, we ran the rest of the algorithms for horizon 1–30 (see Figures 25, 27 and 29). As shown in Figures 24 and 25, the utility is not dramatically affected by the horizon for the optimal, approximations and stop algorithms. This is different for WAIT, MIDDLE and RANDOM, which lose a constant cost every time stamp since they do not intelligently compute where to stop. The stop strategy is not affected by the horizon since it always makes the decision at the first time, thus we see that for low horizon levels a wait strategy is better, but for high levels the stop strategy outperforms the wait strategy as well as MIDDLE and RANDOM.

Although the chain topology of the candidate trees grows linearly with the horizon, the runtime of OPTIMAL increases exponentially in the horizon, since at each level it splits the possible assignments of the candidates' timed variables (Figure 26, the runtime is presented in a logarithmic scale). As shown in Figure 27, OPTIMISTIC and PESSIMISTIC grow polynomially but OPTIMISTIC grows faster. Naturally, the relative depth of the decision of OPTIMAL and both approximations decrease in the horizon, since the cost function grows in the horizon and thus it is less worthwhile to wait for more information (Figure 28 and 29). Nevertheless, the decision time does not converge to 0 since once the cost is not very high, the cost of waiting a few time stamps may be less than the expected utility. Obviously, as discussed above, the higher the cost the less the wait. Note that

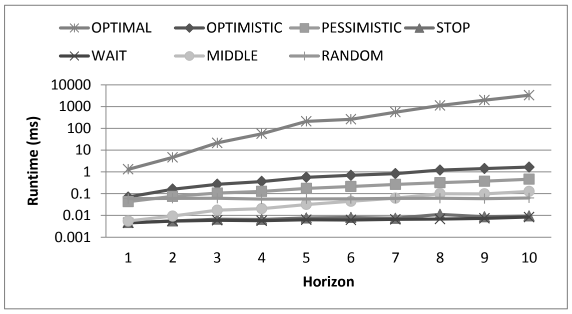


Figure 26: Runtime over horizon (1-10) where number of candidates is 5.

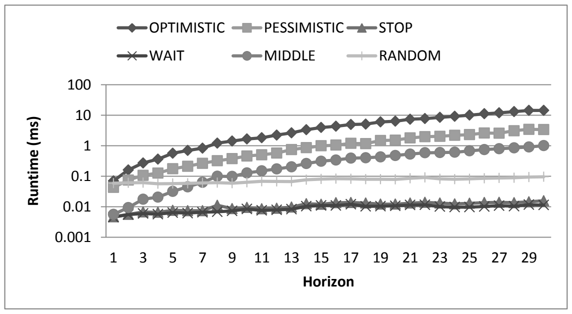


Figure 27: Runtime over horizon (1-30) where number of candidates is 5.

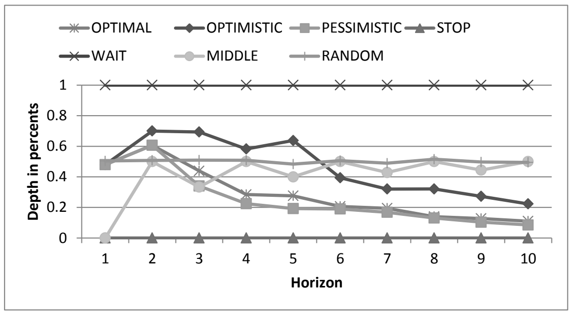


Figure 28: Depth of decision over horizon (1-10) where number of candidates is 5.

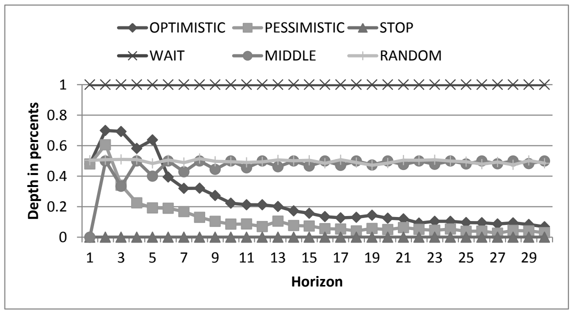


Figure 29: Depth of decision over horizon (1-30) where number of candidates is 5.

the decrease of the OPTIMISTIC compared to the PESSIMISTIC is moderate (Figure 29) since the approximation error of the expected wait of OPTIMISTIC is greater than that of PESSIMISTIC.

6.7 The Effect of the Utility Distribution

In the above experiments, we simulated the utilities by taking them from a uniform distribution. To simulate a varied range of domains we present additional results where the utilities are taken from a Beta distribution with symmetric and asymmetric cases. A Beta distribution with $\alpha = \beta$ provides a symmetric distribution. For $\alpha = \beta \geq 2$ a Beta distribution is similar to a normal distribution defined in an interval of $[0, 1]$ and thus reflects the distribution of many real-world domains. The larger the value of $\alpha = \beta$ the lower the variance. Running experiments with a Beta distribution allows us to examine: (1) the influence of the variance - by controlling the value of $\alpha = \beta$, and (2) the influence of the skewness - by setting α to a fixed value and varying β.

In the first experiment we set $\alpha = \beta = 2$ and the number of candidates and the horizon at 5. By comparing the results (Figure 31) to the results of the experiments with uniform distribution (Figure 16), we can see that all the algorithms except for WAIT, improve the utility. This can be explained by the fact that the variance between the candidates' utilities in a Beta distribution is smaller than the variance in a uniform distribution, and thus by choosing a non-optimal candidate, the utility is

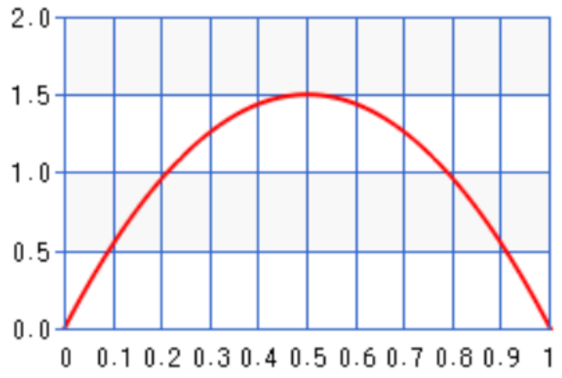


Figure 30: A Beta distribution with $\alpha = 2$ and $\beta = 2$.

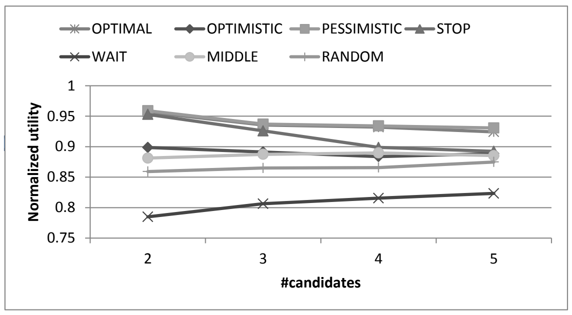


Figure 31: Utilities were taken from a Beta distribution with $\alpha = 2$ and $\beta = 2$ skewness=0. Normalized utility over candidates with a time horizon of 5 levels and 5 candidates.

closer to that of the optimal candidate. Therefore, if an algorithm stops and selects a non-optimal candidate, the utility by such a selection is closer to the optimal (and higher) in a Beta distribution than in uniform distribution. This also explains why in a Beta distribution the utility of STOP is higher than that of OPTIMISTIC, while in a uniform distribution it is lower. This is also supported by the depth of the decision. The decision is made earlier in a Beta distribution than in a uniform distribution. The utility of the WAIT strategy is the same in both distributions because once the decision is made at the end it is optimal and thus only the cost is reduced, which is the same in both distributions.

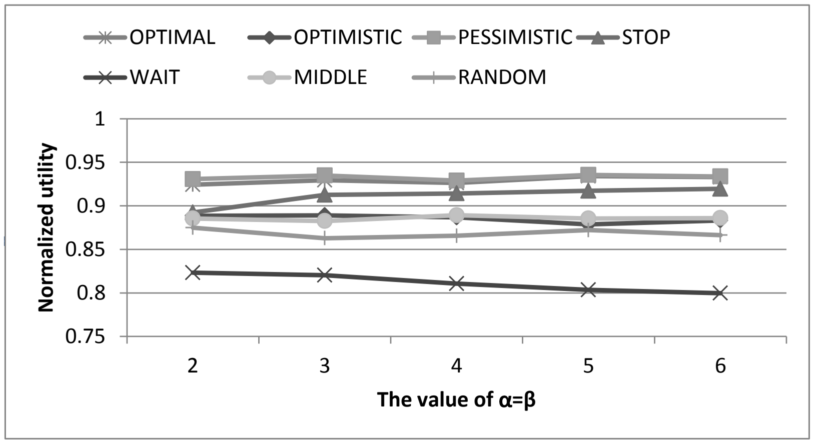


Figure 32: Utilities were taken from a Beta distribution with varied $\alpha = \beta$. Normalized utility over candidates with a time horizon of 5 levels and 5 candidates.

To further examine this insight we run experiments with a varied range of $\alpha = \beta$ of 2 to 6. By increasing α and β the variance is decreased. Figure 32 presents the average utility for this experiment. It seems that the utility of most algorithms is not significantly influenced by the variance. A possible explanation is the tradeoff between two trends. On the one hand, the smaller the variance between the utilities of the candidates, the less difference between the utilities of the chosen candidate and the best candidate. On the other hand, the difference between the expected utility of the candidates, decreases with the variance, and thus the possibility of selecting a wrong candidate increases. Consequently, the probability for an error in choosing the best candidate increases but the payoff for an error is decreased and thus the utility is not significantly influenced by the variance. An exception is the WAIT algorithm which decreases with the variance. The reason is that it makes the decision only at the end and thus always chooses the best candidate. However, since a Beta distribution with $\alpha = \beta$ is a symmetric distribution then as the variance increases the best utility decreases (close to the middle), and on the other hand, it pays a full cost for waiting to the last time stamp.

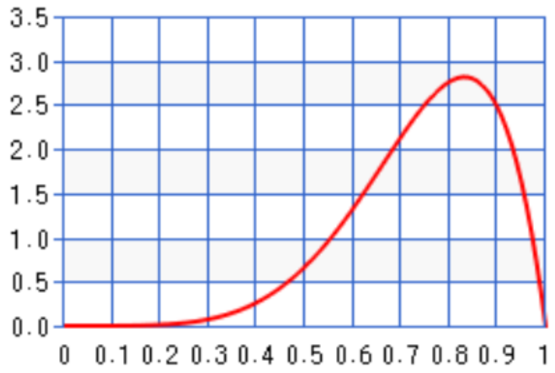


Figure 33: A Beta distribution with $\alpha = 2$ and $\beta = 6$.

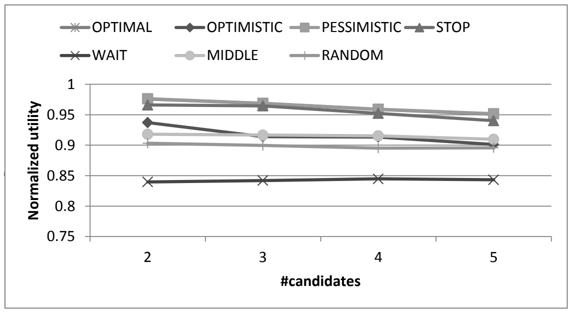


Figure 34: Utilities were taken from a Beta distribution with $\alpha = 2$ and $\beta = 6$ skewness=0.69. Normalized utility over candidates with a time horizon of 5 levels and 5 candidates.

We repeated the experiments with a Beta distribution while changing the parameters of the distribution only for the first candidate to $\alpha = 6$ and $\beta = 2$. This difference influences the skewness of the distribution (-0.69) and gives a high probability of gaining higher values (Figure 33). Since α and β parameters of the other candidates remain the same ($\alpha = 2$ and $\beta = 2$), their skewness is 0 and thus the first candidate is likely to be chosen. As shown in Figure 34, low skewness of the first candidate significantly increases the utility of all the algorithms. The reason is that independently of the stopping time, in most cases, the expected utility of the first candidate is the highest and thus it is selected by the algorithms. Then only the cost of waiting reduces the utility. This insight is significantly shown in the high utility of the STOP algorithm.

We further experimented with the influence of the skewness on the algorithms. Figure 35 presents the normalized utility by changing the skewness of the first candidate. The lower the skewness, the more likely the first candidate will be chosen. The increase in the utility of all algorithms is clear since the more likely the first candidate will be chosen, the fewer errors there will be in choosing the best candidate.

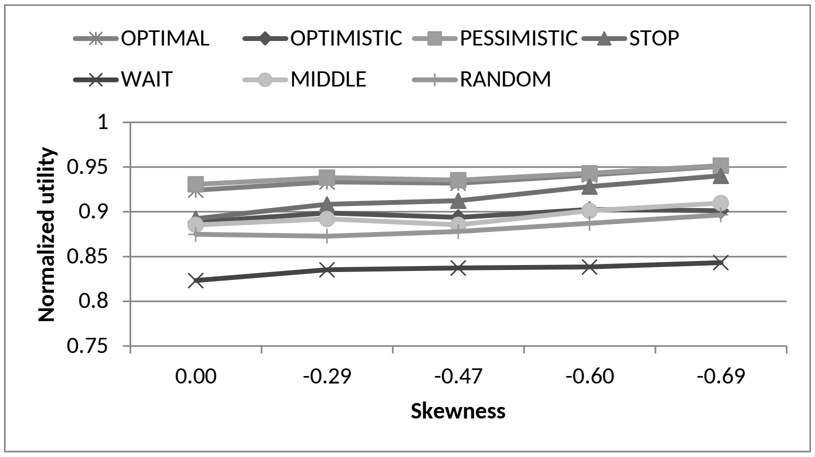


Figure 35: Utilities were taken from a Beta distribution with $\alpha = 2$ and varied β and skewness. Normalized utility over candidates with a time horizon of 5 levels and 5 candidates.

6.8 Conclusions

To summarize, the conclusions from the experiments are:

1. As the cost function increases more moderately (a root function), the PESSIMISTIC algorithm becomes less effective and the OPTIMISTIC becomes better than the PESSIMISTIC.

2. The utility of PESSIMISTIC is very close to OPTIMAL and in most cases (for functions that grow polynomially) there is no statistically significant difference between them.

3. The runtime of OPTIMAL is exponential and actually feasible only for a few candidates with small candidate trees.

4. The runtime of the approximations is polynomial but PESSIMISTIC runs much faster than OPTIMISTIC.

5. There is no statistically significant difference between the experiments that include shared timed variable and experiments that include only disjoint timed variables (except for MIDDLE and RANDOM).

6. OPTIMAL runs much faster in experiments which include candidates with shared timed variable.

7. PESSIMISTIC strategy makes the decision slightly earlier than OPTIMAL.

8. OPTIMISTIC strategy makes the decision much later than OPTIMAL. The gap is increased by increasing the horizon of the candidate trees and the cost of waiting. For a very large horizon and cost the depth is very close to time=0.

9. There are cases (low horizon, high number of agents, cost functions that grow polynomially and shared timed variables) where MIDDLE is better than the OPTIMISTIC algorithm.

10. The greater the expected utility of one candidate is higher than the others, the greater the utility achieved by the different algorithms.

11. The normalized utilities achieved by the algorithms is almost not affected by the variance of the candidates' utilities. However, a significant improvement is achieved with a Beta distribution ($\alpha = \beta \geq 2$) in relation to uniform distribution.

7. Related Work

In this section we discuss the relation between this work and other research related to the optimal stoping problem and exploration—exploitation problems.

7.1 Optimal Stopping Problem

Our problem is related to **the Optimal Stopping Problem (OSP)**. In OSP the goal is to choose a time to take a particular action in order to maximize the expected reward (Ferguson, 1989; Gilboa & Schmeidler, 1989; Peskir & Shiryaev, 2006). The classical stopping problem is defined by two objects: (i) a sequence of independent random variables, $X_1, X_2, \ldots$, with a known joint distribution, and (ii) a sequence of real-valued reward functions, $y_0, y_1(x_1), y_2(x_1, x_2), \ldots$,. For each $n = 1, 2, \ldots$, after observing $X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n$, an agent may stop and receive the known reward, $y_n(x_1, \ldots, x_n)$, or it may continue and observe X_{n+1}. If the agent chooses not to make any observation, it will receive the constant amount, y_0. Take for example the "house-selling problem" where an agent wishes to sell a house. Each day it receives an offer X_i. The agent should decide either to accept the offer or to wait for the next offer. Waiting is associated with a cost of living. Offers are assumed to be independent. The goal is to get the highest offer (Lippman & McCall, 1976).

This class of problems seems to be similar to our problem, since they both address the problem of finding the best time to stop. However, in a deeper perspective, there are three significant differences between these problems:

(1) Stopping reward: In OSP, an agent has to decide whether to stop and obtain *a known reward, based on the prior random variables* or to wait until the next time stamp and observe the next random variable. If the agent chooses to stop, and not to make any observation, it will receive a constant reward that is not dependent on further random variables. In contrast to OSP, in our model, the utility (which is not considering the waiting cost) of each candidate does not depend on the decision to stop or wait, but on future events. Even the gain is not solely dependent on the stopping time, since different future events influence the gain differently. A waiting decision enables the decision maker to acquire more information about the reward of each candidate, although this reward is not affected by the waiting decision. For example in the house-selling problem, an agent wishes to sell a house. Each day it receives an offer X_i. The agent should decide either to accept the offer or to wait for the next offer. If he decides to stop and accept the offer he obtains the reward of the offer. This reward is not affected by the future offers. In our model, on the other hand, the reward from stopping depends on future random variables. This difference in the reward obviously affects the way each approach maximizes the reward. For example, assume an omniscient agent that knows the outcomes of all the variables in advance; it will certainly stop at the time stamp with the highest value in the OSP. Contrastingly, in our problem, the best time to stop is at the first time stamp since at that time the agent knows the exact utility of the candidates rather than the expected utility.

The second aspect is related to the independency of the stopping rewards. In OSP, the stopping rewards of each time stamp are assumed to be independent. For example, in the house-selling problem, the offers are independent. In our problem, although the variables are independent, the rewards from stopping are dependent. This difference is significant since, with the independency assumption, the stopping rule of OSP depends on the probability that the agent did not stop until the current time stamp multiplied by the probability that it will stop now. In our model, on the other hand, the rewards from stopping depend on future random variables and are calculated by taking the expected rewards of the candidates and are therefore dependent. As a result, we are not able to use the OSP model to solve our problem and vice versa.

(2) Multiple candidates: In OSP there is one reward from observing $X_1, X_2, ..., X_n$. Our problem, on the other hand, considers multiple candidates. As a result, each candidate has a different reward from observing the variables. There are two different challenges that we face: (1) finding the best time to stop, and (2) choosing the candidate with the highest expected utility at that time. Although these are two different challenges, they cannot be treated in two steps: finding the best time to stop in advance, and when that time is reached choosing the candidate with the highest expected utility at that time. Such an approach would have made the challenge much simpler and thus the fact that there are multiple candidates would be insignificant. Nevertheless, the problem cannot be solved in two separate steps since the decision of whether to stop or wait depends on the assignments that will occur the next time. Since each candidate may be affected by different assignments, we should consider the combination of all assignments which makes this problem hard.

(3) Joint distribution: In the classical stopping problem the random variables have a known joint distribution. In the previous example, all the offers are assumed to have the same known distribution and thus in case of infinity there is convergence. Our problem is different since it considers random variables with different distributions.

In the last few years several researchers have generalized the classical stopping problem in order to deal with cases of multiple distributions, i.e., multiple optimal stopping problems. Riedel (2009) presents a unified and general theory of optimal stopping under multiple priors in discrete time and extends the theory to continuous time (da Rocha & Riedel, 2010; Cheng & Riedel, 2010). He developed a theory of optimal stopping with more than one joint distribution and with unknown distribution of the variables using and extending suitable results from the martingale theory (Williams, 1991). Still the two differences (Stopping reward, Multiple candidates) specified above, are maintain. In addition, Riedel presents specific constraints on the random variables such as considering the Martingale theory and assuming that the set of stopping rewards are time-consistent (Cheridito & Stadje, 2009). In contrast, in our work the distribution of the variables is known in advance and we do not require any specific constraints on the random variables.

These three differences demonstrate that in spite of the similarity between our problem and OSP, these two models are not comparable and cannot be reduced to each other.

7.2 Exploration—Exploitation Problems

In addition to the classical stopping problem, we also consider a subset of the family of **exploration—exploitation problems** as a kind of stopping problem. In these problems an agent has to decide when to stop acquiring information (exploration) about a specific issue and make a decision (exploitation). **Sequential hypothesis testing** is a method which is based on statistical tests. This method enables a stopping rule to be defined as soon as significant results are observed. This method is based mainly on uncertain information, using multiple observations and samples. For instance, Wald and Woldforwitz (1948) present a problem where a chance variable, X's distribution, only can be either $p_0(X)$ or $p_1(X)$. The required decision is to choose between the two options based on acquired observations of this variable. The research goal obviously, is to make a decision with a minimal number of observations.

In the **multi-armed bandit problem** (Katehakis & Veinott, 1987) an agent allocates trials over slot machines where each machine provides a random reward from a distribution specific to that machine. The objective is to allocate trials over the slot machines in order to maximize the expected reward from using the machines. One version of the multi armed bandit which is the most relevant to our research is the Max K-Armed Bandit. In the Max K-Armed Bandit problem (Cicirello & Smith, 2005; Streeter & Smith, 2006) the objective is to allocate trials over the K arms in order to identify the best arm. Cicirello and Smith (2005) extend the problem where the agent runs trials repeatedly, where in each trial it tries to improve the reward it has achieved thus far.

Similarly, in the **ranking and selection problem** an agent is supposed to select an alternative among several options. To demonstrate the problem, Powell and Ryzhov present the following example (Powell & Ryzhov, 2012): a physician should choose a type of drug among several medicines which reduces the cholesterol of a patient,. In order to decide about the best drug he might require making some physical experiments or he might need to run a number of medical laboratory simulations. Testing an alternative might involve running a time-consuming computer simulation, or require a physical experiment. Obviously, the experimental process is costly and thus the challenge is to allocate the experiments that most efficiently and accurately make the selection. Usually there is a limited budget for evaluating the alternatives and when the budget is exhausted, the agent has to choose the alternative that appears to be the best, according to the obtained knowledge (Swisher, Jacobson, & Yücesan, 2003). Frazier and Powell (2008) present a version of this problem where the prior information units and the new obtained information units about each alternative are sampled from a specific distribution with unknown mean and variance. The model provides a new heuristic sampling and stopping rule that relies on the distribution of the samples.

An additional problem in statistical analysis is **change detection** which tries to identify a change in the parameters of a stochastic system. The changes can be in the probability distribution of a stochastic process or time series (Basseville & Nikiforov, 1993). In general, the problem concerns both detecting whether or not a change has occurred (several changes also might occur), and identifying the time of any such change. The model has to decide when to stop obtaining observations and find the closest time stamp that the distribution has changed. The problem in some works is to detect the disorder time as quickly as possible after it happens and minimize the rate of false alarms at the same time (Dayanik, Poor, & Sezer, 2007).

Another problem is decision making concerning **multiple observations that are informative but expensive**. The challenge with decision making problems is to decide which variables to observe in order to maximize the expected utility. Krause and Guestrin studied this problem in the domain of sensor placement and consider a sensor network where the utility of a sensor is determined by the certainty about the measured quantity. The task is to efficiently select the most informative subsets of observations. Specifically, they propose optimal nonmyopic value of information in chain graphical models (Krause & Guestrin, 2009). Bilgic and Getoor (2011) address a similar problem for efficiently acquiring classification features in domains in which costs are associated with acquisition. The objective is to minimize the sum of the information acquisition cost. They propose a data structure known as the "value of information lattice" (VOILA). VOILA exploits dependencies between missing features, making it possible to share information value computations between different feature subsets.

Similarly, another work (Tolpin & Shimoni, 2010; Radovilsky & Shimoni, 2010) deals with selection under uncertainty and develops algorithms based on the value of information (VOI) with a semi-myopic approximation scheme for problems with real-valued utilities. In particular, Tolpin and Shimoni (2010) interpret VOI as the expected difference between the expected utility of a meta-level action and the expected utility of the current base-level action. Radovilsky and Shimoni (2010) deal with optimizing the selection of a set of observations. Their aim is to bring an objective function to optimum while taking into consideration the cost of observation and the remaining uncertainty after executing the observation.

Recently, Chen et al. (2014) proposed to use the computation of the **Same-decision Probability (SDP)** in order to compute whether additional information should be gathered. In particular, they compute a stopping criterion by computing SDP, the SDP is the probability that the same decision will be made even with further observations. If more information should be gathered they propose which pieces of information to gather next.

Our work is also related, in some aspects, to Horvitz's work (2001, 2013) on **decision making under bounded resources**. The execution of a task is associated with a utility and a cost, depending on resources. When the resources are bounded, the question is which stopping point is the best that will, for the most part, satisfy the task. Horvitz presents the use of an expected value of computation to determine the best time to stop. Similarly, **monitoring anytime algorithms** (Boddy & Dean, 1994; Zilberstein, 1996; Zhang, 2001) search for the best possible answer under the constraint of limited time and/or resources. A major question that arises in utilizing this class of algorithms is how to optimally decide when to stop. For instance, Finkelstein and Markovitch (2001) developed algorithms that design an optimal query schedule to detect when a given goal has been fulfilled. Their aim was to minimize the number of queries (which are time consuming) to reach the goal.

The joint objective of the above works is to maximize a goal function while considering the cost of the observations/acquisition/actions and their extent of uncertainty. We also attempt to maximize the utility by choosing the best candidate and we consider the cost and uncertainty of the timed variables. However, our work differs from the above works in two significant aspects:

1. Time and variables: some of the previous studies such as "change detection problem" and "K-Armed bandit problem" consider a set of observations without considering their order. Most works do not consider the time of the observations. In our work the variables are associated with time and their selection is dynamically determined according to the time progress and the outcomes of the previous variables. This point is important due to the fact that the candidates may be influenced by disjoint variables. We cannot order the variables according to the time and select a subset of variables since the available variables at time t depend on the assignment at time $t-1$. Thus, the decision of whether to stop or wait depends on assignments that will occur at the next time period. Since each candidate may be affected by different assignments we should consider the combination of all assignments which complicates our problem and makes it dissimilar to previous work.

2. Finite and small horizon: in our model the utility of a candidate is affected by a finite and small set of discrete random variables. As a result, the decision maker can actually achieve absolute information about the optimal choice by waiting until the last time stamp. At the last time stamp, there is complete knowledge about the assignments of all the random variables which explore the exact value of the utility. Due to the cost of the information the research problem in our model is to determine the optimal time to make the decision before reaching the end. In the related research, on the other hand, the basic assumption is that only partial information can be obtained and thus it is impossible to compute the exact utility of each candidate. The obtained information contains some observations or samples about the different alternatives that can help the decision maker to statistically approximate the utility distribution of the different alternatives. The potential population of the observations may be infinite or very large and thus it is impossible in practice to obtain all the information necessary to compute the exact utility. The research problem of the above models is thus to use statistical methods to decide on the required information of the different alternatives.

8. Summary and Future Work

In this paper we presented the problem of decision making among multiple candidates where the information arrives dynamically. We focused on the question of when to stop and make a decision that maximizes the utility while taking into consideration the cost of waiting. We presented three algorithms; an optimal algorithm that is exponential in the number of candidates and two alternative polynomial approximation algorithms. We proved that one approximation algorithm is optimistic. Namely its computation of the expected utility from waiting is equal to, or higher than, the expected utility computed by an optimal algorithm, while the other algorithm is pessimistic and thus stops earlier. An empirical evaluation of our algorithms showed that the cost function much influences on the results. As the cost function increases more moderately (a root function), the PESSIMISTIC algorithm becomes less effective and the OPTIMISTIC becomes better than the PESSIMISTIC. For polynomial cost functions there is no significant difference between the outcome utility of the optimal algorithm and the pessimistic algorithm. We also illustrated the exponential growth of the optimal algorithm and the polynomial growth of the optimistic and the pessimistic algorithms.

In the future we plan to continue in two directions: 1) In this work we assumed discrete variables, however in practice, these variables may be continuous. One option for solving this problem is to discretize the variables, however, by doing so we lose optimality. We plan to find an optimal way to address this question, and 2) we plan to further investigate the problem presented in this paper in domains involving multi-agent decision making. In these domains multiple agents should share the same decision based on different variables and utilities. A multi-agent version of our approximation grows exponentially in the number of agents and thus we plan to reduce this complexity.

Appendix A. TDM problem is NP-hard

Proof: We present a reduction from the 3-SAT problem (Cook, 1971). An instance of 3-SAT is given by a propositional logic formula $\Phi(z_1, ..., z_n) = \psi_1 \bigwedge ... \bigwedge \psi_k$, where each ψ_i is a clause with a disjunction of exactly three literals. The aim being to answer "yes" if there is some assignment to the Boolean variables $z_1, ..., z_n$ that satisfies the formula. We construct an instance of TDM as follows.

1. For each Boolean variable z_i we create a timed variable X_i.

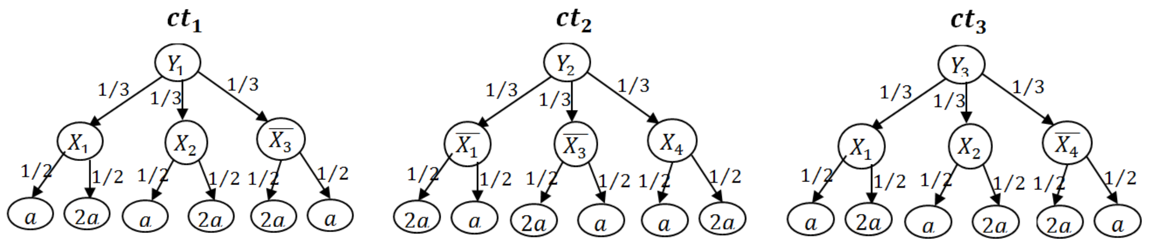


Figure 36: Structure of the candidate trees as accordance with the next 3-SAT formula: $\Phi(z_1, z_2, z_3, z_4) = (z_1 \vee z_2 \vee \neg z_3) \wedge (\neg z_1 \vee \neg z_3 \vee z_4) \wedge (z_1 \vee z_2 \vee \neg z_4)$.

2. For every clause ψ_j $j \in \{1, ..., k\}$ we create a candidate tree ct_j with three timed variables:X_{j_1}, X_{j_2} and X_{j_3}, corresponding to the variables in that clause. For example, for the clause $\psi_j = (z_1 \vee \neg z_4 \vee z_5)$ we create a candidate tree ct_j with variables: X_1, X_4 and X_5.

3. The root of each candidate tree ct_j includes additional timed variable Y_j, where its time stamp is $\Gamma(Y_j) = 1$ with three possible assignments. The probability of each assignment is $\frac{1}{3}$ and each one of the assignments leads to one of the above timed variables:X_{j_1}, X_{j_2} and X_{j_3}.

4. Every timed variable X_i that corresponds to a literal z_i has only two possible assignments $X_i = 1$ with utility $a > 0$ and $X_i = 0$ with utility $2a$, each of them with probability 0.5 and $\Gamma(X_i) = 2$. Each timed variable which is corresponding to a literal $\neg z_i$ has also two possible assignments $X_i = 1$ with utility $2a > 0$ and $X_i = 0$ with utility a, each one of them with probability of 0.5 and $\Gamma(X_i) = 2$.

   Figure 36 presents an example to the structure of the candidate trees as accordance with 3-SAT formula. A left outgoing edge of a random variable X_i represents the assignment $X_i = 1$ and a right outgoing edge represents the assignment $X_i = 0$.

5. We set the time horizon $T = [0, 2]$.

6. We set the cost function to be: $\mathcal{CST}(t) = 0.1 \cdot a \cdot t$.

7. We set a constant C such that $C = 1.8a$.

We now prove that there exists a policy π such that the global expected gain $\mathcal{GEG}(CT, \sigma^0, \pi) \geq C$, if and only if $\Phi(x_1, ..., x_n)$ is not satisfiable.

The expected utility from stopping at time $t = 0$ as well as at $t = 1$ is exactly $\frac{3a}{2}$ which is less than C. But the highest expected utility of $2a$ can be obtained by waiting to time stamp 2. The expected gain is then less than or equal to $1.8a$ (after considering the cost of waiting). Therefore, in the rest of the proof we will consider the waiting policy that waits to time stamp 2.

1. To guarantee $\mathcal{GEG}(CT, \sigma^0, \pi) \geq C$ at time stamp 2, we must confirm that for each assignment and for each combination between the trees branches, at least one candidate tree has a utility of $2a$.

2. By construction, this may happens if and only if for all assignments there is at least one candidate tree in which all its utilities are $2a$.

3. By construction, in all candidate trees an assignment that guarantees a utility of $2a$ entails at least one false clause, meaning all literals in the clause obtain 0 and as a result $\Phi(z_1, ..., z_n)$ is not satisfiable.

As a result we obtain that Timed Decision Making (TDM) is NP-hard.□

References

Basseville, M., & Nikiforov, I. V. (1993). *Detection of abrupt changes: theory and application*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Beaver, B. M., & Mendenhall, W. (1983). *Introduction to probability and statistics, sixth edition, William Mendenhall, study guide*. Statistics Series. Duxbury Press.

Bilgic, M., & Getoor, L. (2011). Value of information lattice: Exploiting probabilistic independence for effective feature subset acquisition. *Journal of Artificial Intelligence Research (JAIR)*, *41*, 69–95.

Boddy, M., & Dean, T. L. (1994). Deliberation scheduling for problem solving in time-constrained environments. *Artificial Intelligence*, *67*(2), 245–285.

Boutilier, C., Dearden, R., & Goldszmidt, M. (2000). Stochastic dynamic programming with factored representations. *Artificial Intelligence*, *121*, 49–107.

Chen, S. J., Choi, A., & Darwiche, A. (2014). Algorithms and applications for the same-decision probability. *Journal of Artificial Intelligence Research (JAIR)*, *49*, 601–633.

Cheng, X., & Riedel, F. (2010). Optimal stopping under ambiguity in continuous time. Working papers 429, Bielefeld University, Institute of Mathematical Economics.

Cheridito, P., & Stadje, M. (2009). Time-inconsistency of var and time-consistent alternatives. *Finance Research Letters*, *6*(1), 40–46.

Cicirello, V. A., & Smith, S. F. (2005). The max *k*-armed bandit: A new model of exploration applied to search heuristic selection. In Veloso, M. M., & Kambhampati, S. (Eds.), *AAAI*, pp. 1355–1361.

Cook, S. A. (1971). The complexity of theorem-proving procedures. In *STOC '71: Proceedings of the third annual ACM symposium on Theory of computing*, pp. 151–158, New York, NY, USA. ACM.

da Rocha, V. F. M., & Riedel, F. (2010). On equilibrium prices in continuous time. *Journal of Economic Theory*, *145*(3), 1086–1112.

Dayanik, S., Poor, H. V., & Sezer, S. O. (2007). Multisource bayesian sequential change detection. *CoRR*, *abs/0708.0224*.

Ferguson, T. S. (1989). Who solved the secretary problem?. *Statistical Science*, *4*(3), 282–289.

Finkelstein, L., & Markovitch, S. (2001). Optimal schedules for monitoring anytime algorithms. *Artificial Intelligence*, *126*, 63–108.

Frazier, P., & Powell, W. (2008). The knowledge-gradient stopping rule for ranking and selection. In *Simulation Conference, 2008. WSC 2008. Winter*, pp. 305–312.

Gilboa, I., & Schmeidler, D. (1989). Maxmin expected utility with non-unique prior. *Journal of Mathematical Economics*, *18*(2), 141–153.

Guestrin, C., Koller, D., Parr, R., & Venkataraman, S. (2003). Efficient solution algorithms for factored MDPs. *Journal of Artificial Intelligence Research (JAIR)*, *19*, 399–468.

Horvitz, E. (2001). Principles and applications of continual computation. *Artificial Intelligence*, *126*, 126–1.

Horvitz, E. (2013). Reasoning about beliefs and actions under computational resource constraints. *CoRR*, *abs/1304.2759*.

Kalech, M., & Pfeffer, A. (2010). Decision making with dynamically arriving information. In van der Hoek, W., Kaminka, G. A., Lespérance, Y., Luck, M., & Sen, S. (Eds.), *AAMAS*, pp. 267–274.

Katehakis, M., & Veinott, J. A. (1987). The multi-armed bandit problem: decomposition and computation. *Mathematics of Operations Research*, *12*(2), 262–268.

Krause, A., & Guestrin, C. (2009). Optimal value of information in graphical models. *Journal of Artificial Intelligence Research (JAIR)*, *35*, 557–591.

Lippman, S. A., & McCall, J. J. (1976). The economics of job search: A survey. *Economic Inquiry*, *14*(3), 155–189.

Peskir, G., & Shiryaev, A. (2006). *Optimal Stopping and Free-Boundary Problems*. Birkhäuser Basel.

Powell, W., & Ryzhov, I. (2012). *Optimal Learning*. Wiley Series in Probability and Statistics. Wiley.

Radovilsky, Y., & Shimoni, S. E. (2010). Observation subset selection for optimization under uncertainty. Tech. rep., Lynne and William Frankel Center for Computer Science at Ben Gurion University of the Negev.

Reches, S., Kalech, M., & Stern, R. (2011). When to stop? that is the question. In Burgard, W., & Roth, D. (Eds.), *AAAI*. AAAI Press.

Riedel, F. (2009). Optimal stopping with multiple priors. *Econometrica*, *77*(3), 857–908.

Streeter, M. J., & Smith, S. F. (2006). An asymptotically optimal algorithm for the max k-armed bandit problem. In *AAAI*, pp. 135–142. AAAI Press.

Swisher, J. R., Jacobson, S. H., & Yücesan, E. (2003). Discrete-event simulation optimization using ranking, selection, and multiple comparison procedures: A survey. *ACM Trans. Model. Comput. Simul.*, *13*(2), 134–154.

Tolpin, D., & Shimoni, S. E. (2010). Semi-myopic observation selection for optimization under uncertainty. Tech. rep. 10-01, Lynne and William Frankel Center for Computer Science at Ben Gurion University of the Negev.

Wald, A., & Wolfowitz, J. (1948). Optimum Character of the Sequential Probability Ratio Test. *Annals of Mathematical Statistics*, *19*(3), 326–339.

Williams, D. (1991). *Probability with Martingales*. Cambridge mathematical textbooks. Cambridge University Press.

Zhang, W. (2001). Iterative state-space reduction for flexible computation. *Artificial Intelligence*, *126*(1-2), 109–138.

Zilberstein, S. (1996). Using anytime algorithms in intelligent systems. *AI Magazine*, *17*(3), 73–83.