Self-triggered optimal servicing in dynamic environments with acyclic structure

Cameron Nowzari Jorge Cortés

***Abstract*—This paper considers a class of scenarios where targets emerge from some known location and move towards some unknown destinations in a weighted acyclic digraph. A decision maker with knowledge of the target positions must decide when preparations should be made at any given destination for their arrival. The decision maker faces a timing trade-off: early decisions mean more time for preparation at the cost of higher uncertainty in the target's true destination while later decisions mean less uncertainty at the cost of having less time to prepare. We show how this problem can be formulated as an optimal stopping problem on a Markov chain. This sets the basis for the introduction of the BEST INVESTMENT ALGORITHM which prescribes when investments must be made conditioned on the target's motion along the digraph. We establish the optimality of this prescription and examine its robustness against changes in the problem parameters, identifying sufficient conditions to determine whether the solution computed by the BEST INVESTMENT ALGORITHM remains optimal. Based on this analysis, we develop the SELF-TRIGGERED ACQUISITION&DECISION ALGORITHM that allows the decision maker, under partial knowledge of the parameter dynamics, to schedule in advance when to check if the control policy in its memory remains optimal and, if this test fails, when to recompute it. Finally, we obtain worst-case lower bounds on the maximum time that can elapse under arbitrary parameter dynamics before the optimal solution must be recomputed. Simulations illustrate our results.**

I. Introduction

This paper considers a scenario where targets appear at a known location and move through an acyclic directed graph to some unknown destination, possibly different for each target. The graph is an abstraction that represents connections available to the targets between points of interest in an environment. A group of sensors deployed over the nodes of the network report the presence of targets to a decision maker. For any given target, it is the job of the decision maker to identify the target's potential destination and make preparations accordingly (for instance by committing some resources to the destination). The earlier these preparations are made, the less resources we need. The decision maker must balance the desire to correctly identify the target's true destination with the amount of resources that must be committed.

This type of decision problem appears in a variety of scenarios including supply chain management, resource allocation, queuing, servicing problems, and pursuit-evasion games on road networks. For example, in queuing, targets can be thought of as heterogeneous tasks that travel through a network of nodes where different skills exist for identifying task features. The edge weights represent the time it takes the corresponding node to examine the task against a specific feature. The task release controller must make a decision on the task type and, based on this decision, assign it to someone. The earlier a task is assigned, the higher the risk that it was assigned incorrectly. Similarly, in supply chain management, targets can be thought of as customer demands that must be met by a specific deadline. The supervisor must make decisions as to how much supply of different products to purchase ahead of time to meet customer demand while overstocking as little as possible. In this scenario, the earlier the purchase of a product is made, the lower the price of the product while the higher the uncertainty in the demand. On the other hand, if the supervisor puts off placing orders to more accurately gauge customer demand, the cost of rushing products to customers may have increased. This setup can again be modeled as a target moving through a graph where nodes represent different customer demands at a specific instances of time before the deadline and edge weights represent elapsed time.

Literature review: The subject matter of the problem considered here is optimal decision making under uncertainty, and has connections with Markov decision processes, optimal stopping, and dynamic programming. A discussion of current techniques and challenges related to optimization problems under uncertainty is documented in [1]. Common to almost all these problems is some sort of stochastic dynamic programming solution, see [2], [3], [4]. For a specific class of utility functions, simpler solutions can be found [5]. Interestingly, the problem we pose can be cast as an optimal stopping problem on a rooted directed tree for which we can find an algorithmic solution that scales with the size of the state space. The works [6], [7], [8] present a broad exposition of optimal stopping problems and their applications. For a specific family of optimal stopping problems on Markov chains, [9] establishes existence of solutions and [10] reviews methods to solve them. We refer to [11], [12] for an exposition of the general discrete time and space problem which also introduce a technique called the Elimination Algorithm. This technique finds the optimal solution faster than standard methods by eliminating states from the search that are guaranteed not to belong to the optimal stopping set.

In the context of sensor networks, [13] considers an optimal stopping problem on a hidden Markov chain where the objective is to detect the presence of a target on a line graph with noisy sensor measurements. A variation is considered in [14], where an additional decision can be made at each timestep to pay for perfect information or not. In the context of optimal investments and task servicing, [15] considers the

The authors are with the Department of Mechanical and Aerospace Engineering, University of California, San Diego, CA 92093, USA, {cnowzari,cortes}@ucsd.edu

problem of finding optimal controls at each timestep given stochastic observations whose objective is to steer the target towards a desired goal; however, the algorithms that find the optimal solutions are usually not scalable with the size of the problem. To this point, some papers such as [16], [17] study heuristic approaches to find suboptimal control policies and reduce computation time. We also make a mention here to robust Markov decision problems, in which the probability distributions themselves are uncertain [18]. The notion of robustness that we consider in this paper is different from that of robust stochastic programming problems in the literature. Normally, as presented in [18], robustness is a term attached to an algorithm and conditions for which it can still find the optimal solution. In this paper, we instead analyze the robustness of solutions, i.e., once the optimal solution to the problem has been found, we determine conditions under which the optimal solution does not change. Finally, our exposition is also connected to the increasing body of work on self-triggered control, see e.g. [19], [20], [21], [22], that reduces computation, decision making, and implementation efforts while still guaranteeing a desired level of performance.

Statement of contributions: We begin by formulating the decision problem under uncertainty described above over a weighted acyclic digraph. We then specify a probabilistic model for target motion on the set of paths of the digraph used by the decision maker and formally define the set of allowable control policies. The decision problem then corresponds to the optimization over the set of control policies of an objective function that encodes the expected net reward associated with investing in a destination at a particular time. Our contributions on this problem are threefold. First, we show that the proposed scenario can be formulated as an optimal stopping problem on a Markov chain. We establish the equivalence of finding the optimal control policy with that of finding the optimal stopping set of this optimal stopping problem. Second, we build on this equivalence to design algorithms that find the optimal and second-to-optimal investment policies and characterize their correctness and time complexities. The BEST INVESTMENT ALGORITHM and SECOND BEST INVESTMENT ALGORITHM are dynamic programming-based strategies that iteratively solve simple sub-problems until the desired policy is found. The knowledge of the second best control policy plays an important role in our third contribution, which is the analysis of the robustness of the optimal solution against changing parameters. Specifically, we obtain explicit bounds on the change of value of the objective function caused by modifying the various parameters of the problem: edge weights, path probabilities, and goal rewards. This allows us to identify a sufficient condition to determine if the optimal solution of the problem remains optimal as the parameters change. The condition we obtain only relies on knowledge of the new parameters and the optimal and second-to-optimal control policies for the original parameters. Based on this analysis, we develop the SELF-TRIGGERED ACQUISITION&DECISION ALGORITHM that allows the decision maker, under partial knowledge of the parameter dynamics, to schedule in advance when to check if the control policy in its memory remains optimal (and if this test fails, when to recompute it). The availability of this algorithm yields computational savings and immediate readiness to the decision maker. Finally, we obtain worst-case lower bounds on the maximum time that can elapse under arbitrary parameter dynamics before the optimal solution must be recomputed. Simulations illustrate our results.

Organization: Section II introduces preliminary concepts and notation. Section III introduces the investment decision problem and Section IV reformulates it as an optimal stopping problem on a Markov chain. Section V presents our best investment decision policies and establishes their correctness. Section VI analyzes the robustness of the optimal solution to changing problem parameters and Section VII builds on this analysis to design self-triggered control strategies. Finally, we gather our conclusions in Section VIII.

II. PRELIMINARIES

This section introduces basic but useful concepts from graph theory and Markov chains. We start with some notational conventions. Let $\mathbb{R}$, $\mathbb{R}_{\geq 0}$, $\mathbb{Z}_{\geq 0}$ be the set of real, nonnegative real, and nonnegative integer numbers, respectively. The cardinality of a set X is denoted by $|X|$. Let $\mathbb{P}(d) = \{(\alpha_1, \dots, \alpha_d) \in \mathbb{R}_{\geq 0} \mid \sum_{i=1}^{d} \alpha_i = 1\}$ be the space of probability vectors in $\mathbb{R}^d$.

A. Graph theory

A *weighted directed graph* (or weighted digraph) is a triplet $G = (V, E, A)$ consisting of a set vertices V, a set of edges $E \subset V \times V$ and an adjacency matrix $A \in \mathbb{R}_{\geq 0}^{|V|\times|V|}$ satisfying $a_{i,j} > 0$ if and only if $(v_i, v_j) \in E$. Edges are directed, meaning that they are traversable in one direction only. The sets of *in-neighbors* and *out-neighbors* of $v \in V$ are respectively

$$\mathcal{N}^{\text{in}}(v) = \{v' \in V \mid (v', v) \in E\},$$
$$\mathcal{N}^{\text{out}}(v) = \{v' \in V \mid (v, v') \in E\}.$$

A vertex v is a *source* if $\mathcal{N}^{\text{in}}(v) = \emptyset$ and a *sink* if $\mathcal{N}^{\text{out}}(v) = \emptyset$. A vertex v is *collapsible* if $|\mathcal{N}^{\text{in}}(v)| = |\mathcal{N}^{\text{out}}(v)| = 1$. We let $\widehat{G}$ denote the *collapsed* digraph of G after performing the following operation until no collapsible vertex exists: remove each collapsible vertex v_j and replace the pair of edges (v_i, v_j), (v_j, v_k) by an edge (v_i, v_k) with weight $a_{i,j} + a_{j,k}$.

A *directed path* p, or in short path, is an ordered sequence of vertices such that any two consecutive vertices in p form an edge in E. For a source $s \in V$, we let $\mathcal{P}(s)$ denote all paths that start at s and end at a sink of the digraph. Given $v \in V$, we let $\mathcal{R}(v)$ and $\mathcal{R}^{-1}(v)$ be the set of *descendants* and *ancestors* of v, respectively. In other words there exists a path from v to all $v' \in \mathcal{R}(v)$ and from all $v' \in \mathcal{R}^{-1}(v)$ to v. Given a path $p = (v_1, \dots, v_m)$, let

$$\mathfrak{S}(p) = \bigcup_{k\in\{1,\dots,m\}} (v_1, \dots, v_k)$$

be the set of all subpaths of p. We define the map last to extract the last vertex of a path p, i.e., $\operatorname{last}(p) = v_m$. The length and weighted length of p are respectively

$$\operatorname{lgth}(p) = |p| - 1 \quad \text{and} \quad \operatorname{lgth}^w(p) = \sum_{k=1}^{m-1} a_{i_k i_{k+1}}.$$

Given a sink g and a path p, let $\operatorname{lngth}^{sw}(p,g)$ denote the weighted length of the shortest path from $\operatorname{last}(p)$ to g,

$$\operatorname{lngth}^{sw}(p,g) = \min\{\operatorname{lgth}^{w}(p') \,|\, p' \text{ path from } \operatorname{last}(p) \text{ to } g\}.$$

A path that starts and ends at the same node is called a *cycle*. An *acyclic digraph* is a digraph with no cycles. An acyclic digraph has a finite number of paths and at least one source and sink. A *rooted tree* is an acyclic digraph with a root such that there exists a unique path from the root to each vertex. Each vertex but the the root has exactly one in-neighbor.

B. Optimal stopping problems on Markov chains

Here we introduce optimal stopping problems on discrete Markov chains. Let X be a finite state space and $P \in \mathbb{R}_{\geq 0}^{|X|\times|X|}$ be a row-stochastic matrix. A *Markov chain* starting from $x_0 \in X$ is a sequence of random variables $\{x_k \,|\, k \in \mathbb{Z}_{\geq 0}\}$, where given state x_k at time k, the probability that the state is x_{k+1} at time $k+1$ is $P_{x_k,x_{k+1}}$. The *optimal stopping problem* is a triplet $M = (X, P, Q)$, with X and P as above and $Q : X \to \mathbb{R}$ is a reward function. The value $Q(x)$ is the reward associated with stopping the Markov chain at state x.

Given any $x_0 \in X$, let E_{x_0} denote the expectation of the sequence of random variables $\{x_k \,|\, k \in \mathbb{Z}_{\geq 0}\}$ specified by M and x_0. The goal of the problem is to characterize a set of halting states $Y \subset X$ that maximizes $E_{x_0}[Q(x_\tau)]$, where x_τ is the first time the Markov chain enters Y, i.e., $x_k \notin Y$ for $k < \tau$. A maximizer of this function is an *optimal stopping set* $Y^* \subset X$. Optimal stopping sets can be alternatively characterized in terms of the *value function* $\mathcal{V}^*$,

$$Y^* = \{x \in X \,|\, Q(x) = \mathcal{V}^*(x)\},$$

where

$$\mathcal{V}^*(x_0) = \max_{k\in\mathbb{Z}_{\geq 0}} E_{x_0}[Q(x_k)].$$

III. Problem statement

In this section we introduce the problem of interest. We begin with an informal description of the basic elements and modeling assumptions, then formally describe the problem.

Consider a network of roads described by a weighted acyclic digraph $G = (V, E, A)$ with a single source s and a set of sinks $\mathcal{S}$. Assume targets appear at the source and head towards one of the goals in $\mathcal{S}$, see Figure 1. The weight of an edge can be interpreted as a measure of the cost it takes a target to traverse it (e.g., larger weights correspond to longer times or higher energy cost). Sensors deployed over the graph nodes transmit information about target positions to a decision maker who must decide whether or not to start preparing for the arrival of the target at a goal by committing some resources to it. We refer to this action as 'making an investment.'

Since the destination of each target is unknown, the decision maker must decide when, if ever, to invest in any given goal in anticipation of a target's arrival. Our model specifies that the larger the gap between the investment decision time and the target's arrival time, the less costly it is to make an investment for that goal; however, if an investment is made and the target actually ends up at a different goal, the investment is wasted. Once a decision to invest has been made, it cannot be retracted and thus the cost of investing is incurred immediately.

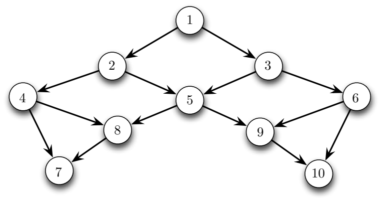


Fig. 1. Example network of roads modeled as a weighted acyclic digraph. All edge weights are equal to 1. There are 8 paths starting at the source (node 1) and ending at a sink (either node 7 or 10). The probabilities associated to these paths are given by the vector $\alpha = [.05, .1, .15, .2, .05, .1, .15, .2]$.

A. Probabilistic model for target motion

The exact way a target T chooses its path along the digraph G is unknown to the decision maker, who instead uses the probabilistic model described next. Let $p_T \in \mathcal{P}(s)$ denote the (unknown to the decision maker) path of T in G. The set of trajectories that can be observed by the decision maker as T moves through G is precisely $\mathfrak{S}(p_T)$. Therefore, the set of all possible trajectories the decision maker may observe is

$$\mathcal{H}(G) = \bigcup_{p\in\mathcal{P}(s)} \mathfrak{S}(p).$$

Let $n = |\mathcal{P}(s)|$ and assign labels $\{1,\dots,n\}$ to the paths in $\mathcal{P}(s)$. Let $\alpha \in \mathbb{P}(n)$ be a probability vector, where α_μ is the probability that target T takes path μ, i.e., $p_T = p_\mu$. Such probabilities can be computed in a number of ways, including incorporating observations about trajectories from past targets, but we do not enter into this here. Note that this model is more general than a Markov chain model on the graph (where the target's motion only depends on its current state).

According to this model, the decision maker can infer a target's future actions as it moves through the network. In fact, given history $h \in \mathcal{H}$, let $\operatorname{Ind}(h) = \{\mu \in \{1,\dots,n\} \,|\, h \in \mathfrak{S}(p_\mu)\}$ denote the set of indices corresponding to the paths that the target could possibly be on. Any path in $\operatorname{Ind}(h)$ is *indistinguishable*, i.e., consistent with having observed the history h. Then, the decision maker can compute the probability that $p_T = p_\mu$ given observation h,

$$\mathscr{P}(p_T = p_\mu |\, h) = \begin{cases} \frac{\alpha_\mu}{\sum_{\nu\in\operatorname{Ind}(h)} \alpha_\nu}, & \text{if } \mu \in \operatorname{Ind}(h), \\ 0, & \text{otherwise.} \end{cases}$$

Using this model, the decision maker can compute the probability that the target will eventually go to a vertex $v \in V$ or another history $h' \in \mathcal{H}$, as follows,

$$\mathscr{P}(v|\, h) = \sum_{\{\mu\in\{1,\dots,n\} \,|\, v\in p_\mu\}} \mathscr{P}(p_T = p_\mu |\, h),$$

$$\mathscr{P}(h'|\, h) = \sum_{\{\mu\in\{1,\dots,n\} \,|\, h'\in\mathfrak{S}(p_\mu)\}} \mathscr{P}(p_T = p_\mu |\, h).$$

Both notions will be used throughout the paper. Note that these will evaluate to 1 if the target vertex or history is already in h and 0 if they are not reachable from h.

B. Allowable control policies

As targets move through the digraph, the decision maker must decide at each timestep, for each goal, whether an investment should be made or not. For simplicity of presentation and without loss of generality, the paper considers investment decisions for one specific goal g (in the case of multiple goals, our policies can then be applied to each one of them). We suppress the dependence on g when it is clear from the context. A control strategy is then a map

$$u : \mathcal{H} \to \{\texttt{invest}, \texttt{not-invest}\}$$

that specifies, for a target with history $h \in \mathcal{H}$, a decision to $u(h)$ in goal $g \in \mathcal{S}$. The set of all allowable control policies is denoted by $\mathcal{U}$. Throughout the paper we consider control strategies that prescribe at most one investment along any given path. The reason for this is that once an investment is made for a goal, it does not make sense to make further investments at the same goal. Formally, for each $p \in \mathcal{P}(s)$, $u(h) = \texttt{invest}$ for at most one $h \in \mathfrak{S}(p)$. If such a history exists, we denote it by $h_u(p)$, otherwise we set $h_u(p) = \emptyset$. We let $\mathrm{Inv}_u = \cup_{\mu \in \{1,\dots,n\}} \{h_u(p_\mu)\}$ be the set of all possible investment histories for policy u.

C. Objective function

Our next step is to define the objective function that the decision maker seeks to optimize. To this end, we present a model for the cost of investment and the reward obtained for making the investment. The cost of investing in goal g for a target with history $h \in \mathcal{H}$ is

$$c(h) = f\Big(\tfrac{1}{\mathrm{lngth}^{sw}(h,g)}\Big),$$

where $f : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is a class $\mathcal{K}$ function, i.e., continuous, monotonically increasing, and satisfies $f(0) = 0$. Note that the longer the weighted length of the shortest path from $\mathrm{last}(h)$ to g, the smaller the cost. The reward for correctly preparing for a target's arrival at g is modeled by a parameter $\beta \in \mathbb{R}_{\geq 0}$. The reward accrued using control policy u is then

$$R_u(p_T) = \begin{cases} \beta, & \text{if } \mathrm{last}(p_T) = g \text{ and } h_u(p_T) \neq \emptyset, \\ 0, & \text{otherwise.} \end{cases}$$

Since the target's path is unknown a priori, this reward is not known when the target is at history $h_u(p_T) \in \mathrm{Inv}_u$. Instead, we define an expected reward using the probabilistic model,

$$E_{h_u(p_T)}[R_u(p_T)] = \beta \mathscr{P}(g|\, h_u(p_T)).$$

Now, maximizing $E_{h_u(p_T)}[R_u(p_T)] - c(h_u(p_T))$ is the job of the decision maker. Since the path of the target is unknown, the *objective function* of the decision problem is then the expected value of this expression over all possible paths,

$$\begin{aligned} J(u) &= E_s\left[E_{h_u(p_T)}[R_u(p_T)] - c(h_u(p_T))\right], \\ &= \sum_{h \in \mathrm{Inv}_u} \mathscr{P}(h|\, s)\left(\beta \mathscr{P}(g|\, h) - c(h)\right). \end{aligned} \qquad (1)$$

IV. Optimal stopping problem formulation

Here, we formulate an optimal stopping problem and establish its equivalence with the decision problem described in Section III. Specifically, for the goal of interest $g \in \mathcal{S}$, we identify a corresponding optimal stopping problem $M_{\text{inv}} = (X, P, Q)$. The advantage of the reformulation is that the target motion is Markov on M_{inv}, whereas in general is not for the original investment problem. The optimal stopping formulation also allows us to establish the existence of an optimal solution and sets the basis for our algorithm design.

A. Optimal stopping problem

According to the probability model for target motion discussed in Section III-A, at any given time, the evolution of a target along G depends on the full history of vertices visited by the target prior to reaching the current vertex. For this reason, we choose as the state space of the optimal stopping problem the set $X = \mathcal{H}(G)$ of all possible target trajectories in G, rather than the set of vertices V. Note that X is a rooted tree with the source s as root. Each node corresponds to a path in G whose unique parent is the subpath obtained by removing the last vertex. The cardinality of X depends on the graph's adjacency matrix A and is upper bounded by

$$|X| \leq 1 + \sum_{p \in \mathcal{P}(s)} \mathrm{lgth}(p),$$

where the summand 1 corresponds to the history s. In the worst case, i.e., when A is a strictly upper triangular matrix containing nonzero elements, one has $|X| = 2^{|V|-1}$.

We define the one-step transition matrix $P \in \mathbb{R}_{\geq 0}^{|X| \times |X|}$,

$$P_{x,y} = \begin{cases} \mathscr{P}(y|\, x), & \text{if } x \in \mathfrak{S}(y), \mathrm{lgth}(y) = \mathrm{lgth}(x) + 1, \\ 0, & \text{otherwise,} \end{cases}$$

for $x, y \in X$. With the definitions of X and P, the target motion now corresponds to a Markov chain on the space of histories with initial condition s. Figure 2 shows the rooted tree X with edge weights given by P for the weighted acyclic digraph depicted in Figure 1.

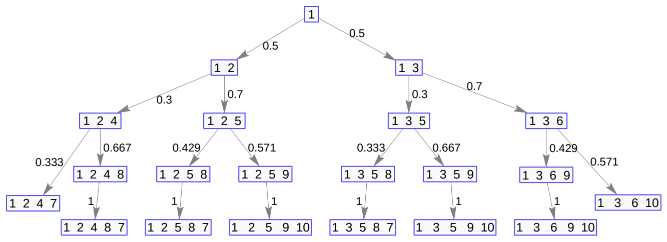


Fig. 2. State space X and transition matrix P of the optimal stopping problem associated to the problem in Figure 1. Each node represents a history $h \in \mathcal{H}$. Note that all the sinks of this tree correspond to a sink of the original digraph.

The last element of the optimal stopping problem is the reward function Q that we define by

$$Q(x) = \beta \mathscr{P}(g|\, x) - c(x). \qquad (2)$$

The first term corresponds to the expected reward obtained from investing in goal g at state $x \in X$ and the second term corresponds to the cost of making this investment.

With the optimal stopping problem $M_{\text{inv}} = (X, P, Q)$ properly defined, the next result follows from [7, Chapter 3] and the fact that X is finite.

Lemma IV.1 (Existence of optimal stopping set) *For $M_{inv} = (X, P, Q)$ constructed as above,*

(i) *there exists an optimal stopping set $Y^* \subset X$, and*
(ii) *no randomized stopping rule can do better than stopping the first time the state is in Y^*.*

B. Equivalence with the decision problem

Here, we establish the equivalence of the optimal stopping problem on a Markov chain M_{inv} with the decision problem described in Section III. To do so, we need a mapping that relates a halting set Y for the optimal stopping problem to a control policy u for the decision problem and vice versa. This is accomplished by introducing the notion of *reduced* halting subset of a halting set Y for a given initial condition x_0 as

$$Y_{x_0} = \{x \in Y \cap \mathcal{R}(x_0) \,|\, y \notin Y \text{ for } y \in \mathcal{R}^{-1}(x)\}.$$

This set is composed of all the states in Y that can be reached first by a Markov chain starting from x_0. In other words, the Markov chain cannot reach states in $Y \setminus Y_{x_0}$ without passing through a state in Y_{x_0}. This set has the property that

$$E_{x_0}[Q(x_\tau)] = E_{x_0}[Q(x_{\tau'})],$$

where x_τ and $x_{\tau'}$ are the first times the Markov chain enters Y and Y_{x_0}, respectively. A halting set Y is *minimal from* x_0 if it satisfies $Y_{x_0} = Y$. To a halting set $Y \subset X$, we then associate the control policy u_Y given by

$$u_Y(x) = \begin{cases} \texttt{invest}, & \text{if } x \in Y_s, \\ \texttt{not-invest}, & \text{otherwise.} \end{cases} \tag{3}$$

Conversely, to a control policy u, we associate the halting set Inv_u. Note that Inv_u is minimal from s because of the defining properties of allowable control policies. We can now draw the connection between the optimal stopping problem and the problem posed in Section III.

Proposition IV.2 (Equivalence with the investment decision problem) *Given an optimal stopping set Y^* for the optimal stopping problem M_{inv}, the control policy u_{Y^*} is optimal for the objective function* (1)*. Reciprocally, given an optimal control policy u^* for the objective function* (1)*, the set Inv_{u^*} is an optimal stopping set that is minimal from s.*

Proof: We proceed by introducing an objective function J_{os} for the optimal stopping problem with initial condition $x_0 = s$. Given a halting set Y, let

$$J_{\text{os}}(Y) = E_s[Q(x_\tau)], \tag{4}$$

where x_τ is the first time the Markov chain enters Y. We can rewrite this function as

$$\begin{aligned} J_{\text{os}}(Y) &= E_s\left[\beta \mathscr{P}(g|\, x_\tau) - c(x_\tau)\right] \\ &= \sum_{x \in Y_s} \mathscr{P}(x|\, s)\left[\beta \mathscr{P}(g|\, x) - c(x)\right]. \end{aligned}$$

With this expression in mind, it is easy to see that

$$J_{\text{os}}(Y) = J(u_Y), \quad J(u) = J_{\text{os}}(\text{Inv}_u),$$

from which the result follows. ∎

The following result is a consequence of Lemma IV.1 and Proposition IV.2.

Corollary IV.3 (Random policies are not better) *No randomized control policy does better than an optimal control policy $u^* : \mathcal{H} \to \{\texttt{invest}, \texttt{not-invest}\}$.*

C. State space reduction for the optimal stopping problem

With the equivalence between the optimal stopping problem $M_{\text{inv}} = (X, P, Q)$ and the decision problem on G established, our strategy to determine the optimal control policy is to find the optimal stopping set Y^*. Before proceeding with the algorithm design, it is advantageous to reduce the size of M_{inv} as this naturally results in lower algorithmic complexities. Our approach proceeds by eliminating states that are guaranteed not to belong to the optimal set. This is reminiscent of techniques such as the Elimination Algorithm [11], [12]. We start by defining a *cluster* $C = (x_1, \dots, x_m)$ as a maximal path in the state space X such that $\mathcal{N}^{\text{out}}(x_k) = \{x_{k+1}\}$ for $k \in \{1, \dots, m-1\}$. Maximal here means that C is not contained in any other path with the same properties. We refer to x_1 as the *anchor* of cluster C. Intuitively, any state in X with only one out-neighbor is part of a cluster with that neighbor. The next result guarantees that eliminating all nodes belonging to clusters other than the anchor nodes does not change the optimal stopping set.

Lemma IV.4 (State space reduction) *Consider the optimal stopping problem $M_{\text{inv}} = (X, P, Q)$. Let $C^1, \dots, C^q$ denote the set of all clusters of X. Then,*

$$Y^* \cap (\cup_{\mu \in \{1,\dots,q\}} C^\mu \setminus \{x_1^\mu\}) = \emptyset.$$

Proof: The result follows by noting that, for any cluster C, the reward obtained by investing at the anchor is higher than the one obtained at any other cluster node, i.e., $Q(x_1) \geq Q(x_r)$ for $r \geq 2$, and thus it is not optimal to stop at x_r. This fact can be established by noticing that, in (2), the expected reward $\beta \mathscr{P}(g|\, x)$ is the same for all $x \in C$ and the cost function c is nondecreasing along the path C. ∎

As a consequence of Lemma IV.4, we define a new optimal stopping problem $\widehat{M}_{\text{inv}} = (\widehat{X}, \widehat{P}, \widehat{Q})$ with state space

$$\widehat{X} = X \setminus (\cup_{\mu \in \{1,\dots,q\}} C^\mu \setminus \{x_1^\mu\}).$$

The transition matrix $\widehat{P}$ is created by modifying the original matrix P. We first replace, for each $\mu \in \{1, \dots, q\}$, the row corresponding to $x_1^\mu \in C^\mu$ by the row corresponding to the last element of C^μ. Then, we remove all rows and columns in P corresponding to the states removed from X. Finally, the reward function $\widehat{Q}$ is just the restriction of Q to $\widehat{X}$. Lemma IV.4 guarantees that the optimal solution found on M_{inv} is the same solution found on $\widehat{M}_{\text{inv}}$. Interestingly,

$$\mathcal{H}(\widehat{G}) = \widehat{X} \cup \mathcal{P}(s),$$

i.e., the space of histories corresponding to the collapsed digraph $\widehat{G}$ is the same as the reduced state-space $\widehat{X}$ plus the set of full paths to the sinks $\mathcal{P}(s)$. A further simplification can be done by eliminating the set $\{x \in \widehat{X} \,|\, g \notin \mathcal{R}(x)\}$ containing the states that do not have the goal g in its set of descendants since their associated reward is trivially zero. Figure 3 illustrates this discussion and the reduction from X to $\widehat{X}$.

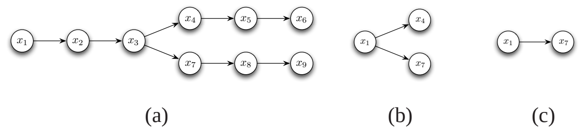


Fig. 3. The original state space X with source $s = x_1$ and sinks x_6 and x_9 is shown in (a). In this case there are $q = 3$ clusters, $C^1 = \{x_1, x_2, x_3\}$, $C^2 = \{x_4, x_5, x_6\}$, and $C^3 = \{x_7, x_8, x_9\}$. The state-space reduction gives rise to $\widehat{X}$ in (b) with only 3 states. If the goal of interest is x_9, the state x_4 can also be removed because x_9 is not reachable from x_4. This can further reduce the state space to a size of two states as shown in (c).

V. Optimal investment decision policies

In this section, we design strategies to find the best and second best control policies for the investment problem in Section III using dynamic programming techniques on the optimal stopping problem formulated in Section IV. The determination of the second best control policy will be useful later in our robustness analysis. The algorithms can be run on either $M = M_{\text{inv}}$ or $M = \widehat{M}_{\text{inv}}$.

A. Best investment algorithm

Here, we present an algorithm to solve the optimal stopping problem $M = (X, P, Q)$ with initial condition s. According to Bellman's principle of optimality [2], the decision at any given state x must be computed assuming that subsequent decisions constitute an optimal policy with respect to this state x and decision $u(x)$. This means before we can find the optimal decision for the source $u^*(s)$, we must already know the optimal solution at all other states. Our algorithm thus starts at states for which the optimal solution can be computed with only one comparison: is it better to invest or wait at this point? These sub-problems can be solved for any state x'_0 once the sub-problems have been solved for all $x \in \mathcal{R}(x'_0)$. Our algorithm iteratively solves sub-problems for each initial condition x'_0 and makes future decisions based on these solutions. The algorithm runs until the problem is solved for s. We now describe the algorithm informally.

> *[Informal description]:* Choose $x \in X$ such that the problem is unsolved for x but solved for its descendants. Compute the value obtained if the chain is stopped at x and compare it to the expected value obtained by waiting one timestep. Save the best decision, store the value, and mark x as solved. Proceed iteratively until the problem is solved for $x = s$.

The strategy is presented formally in Algorithm 1. The next result shows that the output of the BEST INVESTMENT ALGORITHM is the control policy u^*, where $u^*(x) = \texttt{invest}$ for all $x \in Y^*_s$ and $u^*(x) = \texttt{not-invest}$ otherwise.

Proposition V.1 (Correctness of the BEST INVESTMENT ALGORITHM) *Given the optimal stopping problem $M = (X, P, Q)$ for goal g with initial condition s, the* BEST INVESTMENT ALGORITHM *finds the optimal solution u^* to the decision problem and its value $\mathcal{V}^*(s)$.*

Proof: With the notation of the BEST INVESTMENT ALGORITHM, given a state x, the value $V(x)$ corresponds to the value of the objective function obtained by running control policy u_Y on the sub-problem with initial condition x. Since u_Y is constructed from Y, from the proof of Proposition IV.2 we know that $J(u_Y) = J_{\text{os}}(Y_s)$. Therefore, we deduce that $V(s) = J(u_Y)$. Since $\mathcal{V}^*(s)$ is the maximum value of the objective function (4), in order to show that u_Y is a maximizer of (1), all we need to do is establish that $V(s) = \mathcal{V}^*(s)$.

Algorithm 1: BEST INVESTMENT ALGORITHM

```
Initialization:
 1: initialize $V(x) = 0$ for all $x \in X$
 2: initialize Solved $= \emptyset$
 3: initialize $Y = \emptyset$
Perform:
 1: while there exists $x \notin$ Solved with $\mathcal{R}(x) \subseteq$ Solved do
 2:     if $Q(x) \geq \sum_{y \in \mathcal{N}^{\text{out}}(x)} V(y)P_{x,y}$ then
 3:         add $x$ to $Y$
 4:         set $V(x) = Q(x)$
 5:     else
 6:         set $V(x) = \sum_{y \in \mathcal{N}^{\text{out}}(x)} V(y)P_{x,y}$
 7:     end if
 8:     add $x$ to Solved
 9: end while
10: compute $u_Y$
```

We start by verifying that V as determined by Algorithm 1 satisfies the Bellman equation [2],

$$V(x) = \max\{Q(x), \sum_{y \in \mathcal{N}^{\text{out}}(x)} V(y)P_{x,y}\}.$$

This property is enforced by the *if* condition in step 2: of Algorithm 1 for all $x \in X$.

Let $y \in X$ be such that $\mathcal{R}(y) = \emptyset$. Then, it is trivial to see that $V(y) = \max\{Q(y), 0\} = \mathcal{V}^*(y)$ because no other investment decisions will be made after state y. Now, let $(x_0, \dots, x_m)$ be a path ending in $x_m = y$. Then, using backward induction and the Bellman equation, we deduce

$$V(x_{k-1}) = \max\{Q(x_{k-1}), E_{x_{k-1}}[V(x_k)]\} = \mathcal{V}^*(x_{k-1}),$$

for $k \in \{1, \dots, m\}$, which concludes the result. ■

Figure 4 shows the result of an execution of the BEST INVESTMENT ALGORITHM.

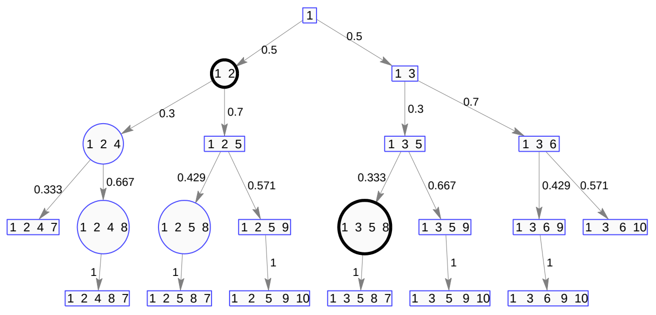


Fig. 4. Optimal solution to the problem described in Figure 1 for the goal of interest g at Node 7, with $\beta = 20$, and cost function $f(z) = 10z$. The optimal stopping set Y^* is depicted by the 5 circular nodes and the minimal optimal stopping set from the source Y^*_s giving rise to the control policy u^* corresponds to the bold circles.

The time complexity of the BEST INVESTMENT ALGORITHM is characterized by the following result. Its proof follows from the fact that the strategy solves $|X|$ sub-problems and each sub-problem is solved in time complexity $O(1)$.

Lemma V.2 *The time complexity of the* BEST INVESTMENT ALGORITHM *to solve M is $O(|X|)$.*

Remarkably, for fixed parameters α, β, and A, Algorithm 1 only needs to be called once to determine a set of rules to follow. Then, without further computations, the decision maker can make decisions depending on the target's position.

B. SECOND BEST INVESTMENT ALGORITHM

Here, we make use of the solution computed by the BEST INVESTMENT ALGORITHM to find the second best solution to the optimal stopping problem M. Our strategy relies on the observation that the optimal stopping set and the second best stopping set are similar. Given an optimal stopping set Y^*, we create *candidate stopping sets*

$$\mathcal{C}(Y^*, x) = \begin{cases} Y^* \setminus \{x\}, & \text{if } x \in Y_s^*, \\ (Y^* \cup \{x\}) \setminus \mathcal{R}^{-1}(x), & \text{otherwise.} \end{cases}$$

Recalling that (3) relates halting sets to control policies, these sets are constructed such that $u_{\mathcal{C}(Y^*,x)}(x) \neq u_{Y^*}(x)$. For simplicity, we let $u_x^{\mathcal{C}} = u_{\mathcal{C}(Y^*,x)}$. The set of all candidate control policies that we search over is then given by $\mathcal{U}^{\mathcal{C}} = \cup_{x \in X}\{u_x^{\mathcal{C}}\}$.

We can now describe the algorithm informally.

[Informal description]: Given the optimal stopping set, create a set of candidate control policies as described above. Select the control policy in this set that has the highest value of the objective function.

We refer to this strategy as the SECOND BEST INVESTMENT ALGORITHM and formally present it in Algorithm 2. The next result shows that the output is the second best control policy.

Algorithm 2: SECOND BEST INVESTMENT ALGORITHM

```
Initialization
 1: initialize $V_x^{\mathcal{C}} = 0$ for all $x \in X$
 2: execute the BEST INVESTMENT ALGORITHM
 3: compute $Y_s^*$ from $Y^*$
Perform:
 1: for $x \in Y_s^*$ do
 2:     set $V_x^{\mathcal{C}} = \mathcal{V}^*(s) - \mathscr{P}(x|\, s)[\mathcal{V}^*(x) - \sum_{y \in \mathcal{N}^{\text{out}}(x)}[\mathcal{V}^*(y) P_{x,y}]]$
 3:     while $\mathcal{N}^{\text{in}}(x) \neq \emptyset$ do
 4:         set $y = \mathcal{N}^{\text{in}}(x)$
 5:         set $V_y^{\mathcal{C}} = \mathcal{V}^*(s) - \mathscr{P}(y|\, s)\,[\mathcal{V}^*(y) - Q(y)]$
 6:         set $x = y$
 7:     end while
 8: end for
 9: compute $\bar{x} = \operatorname{argmax}_{x \in X} V_x^{\mathcal{C}}$
10: compute $u_{\bar{x}}^{\mathcal{C}}$
```

Proposition V.3 (Correctness of the SECOND BEST INVESTMENT ALGORITHM) *Given the optimal stopping problem $M = (X, P, Q)$ for goal g with initial condition s, the* SECOND BEST INVESTMENT ALGORITHM *finds the second best control policy u' to the decision problem, i.e., for all u different from u^* and u',*

$$J(u^*) \geq J(u') \geq J(u).$$

Proof: Note that the value $V_{\mathcal{C}}^x$ in Algorithm 2 is precisely the value obtained by the control policy $u_x^{\mathcal{C}}$, i.e., $V_{\mathcal{C}}^x = J(u_x^{\mathcal{C}})$. Since this algorithm constructs and searches over the policies in $\mathcal{U}^{\mathcal{C}}$, we show here that for every $u \neq u^*$, there exists $u_x^{\mathcal{C}} \in \mathcal{U}^{\mathcal{C}}$ such that $J(u_x^{\mathcal{C}}) \geq J(u)$.

Let x_μ^u be the state $x_\mu^u \in \text{Inv}_u$ at which an investment is prescribed by control policy u along path p_μ. If it exists, we let $Q_\mu^u = Q(x_\mu^u)$, otherwise we let $Q_\mu^u = 0$. Then, given u,

$$J(u^*) - J(u) = \sum_{\mu=1}^{n} \alpha_\mu (Q_\mu^{u^*} - Q_\mu^u),$$

which is trivially nonnegative.

Take now any $\nu \in \{1, \ldots, n\}$ such that $x_\nu^{u^*} \neq x_\nu^u$. Note that there exists at least one such ν because $u \neq u^*$. If the control policy u prescribes an investment along path p_ν later than the optimal investment policy, then we write

$$J(u_{x_\nu^u}^{\mathcal{C}}) - J(u) = \sum_{\mu \in \{1,\ldots,n\} \setminus \text{Ind}(x_\nu^{u^*})} \alpha_\mu (Q_\mu^{u^*} - Q_\mu^u) + \sum_{\mu \in \text{Ind}(x_\nu^{u^*})} \alpha_\mu \left[\left(\sum_{x \in \mathcal{N}^{\text{out}}(x_\nu^{u^*})} \mathscr{P}(x|\, x_\nu^{u^*}) \mathcal{V}^*(x) \right) - Q_\mu^u \right],$$

which is greater than or equal to 0. The first sum is for all the paths at which the optimal investment policy is used compared to the policy u, and thus is easily shown to be nonnegative. The second sum is for all paths going through state $x_\nu^{u^*}$. Investing at $x_\nu^{u^*}$ is optimal; however, u' skips it and uses the optimal policy from the next step onwards. It is also easy to see that this is nonnegative because $x_\mu^u \neq x_\nu^{u^*}$ for all $\mu \in \text{Ind}(x_\nu^{u^*})$.

If the control policy u prescribes an investment along path p_ν earlier than the optimal investment policy, then we write

$$J(u_{x_\nu^u}^{\mathcal{C}}) - J(u) = \sum_{\mu \in \{1,\ldots,n\} \setminus \text{Ind}(x_\nu^u)} \alpha_\mu (Q_\mu^{u^*} - Q_\mu^u),$$

which is simply the optimal solution along all paths that do not go through x_ν^u compared against u, and again can easily be shown to be nonnegative. Thus, $J(u_{x_\nu^u}^{\mathcal{C}}) \geq J(u)$. ■

Figure 5 shows the result of an execution of the algorithm.

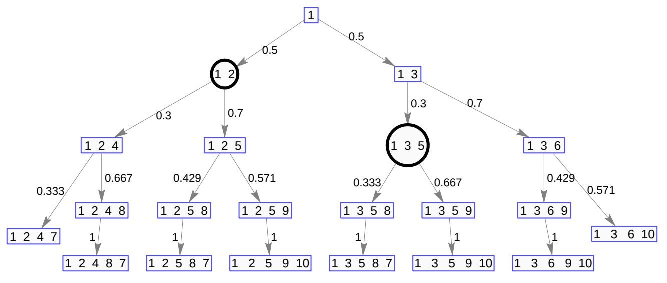


Fig. 5. Second best solution to the problem described in Figure 1 for the goal g at Node 7, with $\beta = 20$, and cost function $f(z) = 10z$. The set of investment states $\text{Inv}_{u'}$ for control policy u' is depicted by the two circular nodes. The values of the objective function for the optimal and second best control policies are given by $J(u^*) = 4.0$ and $J(u') = 3.75$, respectively.

Since the SECOND BEST INVESTMENT ALGORITHM terminates after computing at most $|X|$ candidate values $J(u_x^{\mathcal{C}})$, where each computation has time complexity $O(1)$, we deduce the following statement.

Lemma V.4 *The time complexity of the* SECOND BEST INVESTMENT ALGORITHM *to solve M is $O(|X|)$.*

Remark V.5 (Problem reduction may not preserve the second best solution) Interestingly, although the optimal solutions on M_{inv} and $\widehat{M}_{\text{inv}}$ are the same (cf. Lemma IV.4), this does not hold in general for the second best solution, i.e., the output of the SECOND BEST INVESTMENT ALGORITHM may be different depending on whether it is executed for M_{inv} or $\widehat{M}_{\text{inv}}$. This is because the reduction from M_{inv} to $\widehat{M}_{\text{inv}}$ removes some solutions in M_{inv} that are guaranteed not to be optimal (possibly including the second best policy). As we show later in Remark VII.6, this fact has positive implications on our analysis of the robustness of solutions. •

VI. ROBUSTNESS OF THE OPTIMAL INVESTMENT POLICY

In this section we are interested in determining conditions under which the optimal control policy remains optimal for parameters other than the original ones. Specifically, we consider changes in the edge weights of the digraph, the probability model for target motion, and the reward associated with the goal of interest. Our study is motivated by the idea that small changes in the parameters may not affect the optimal solution, and thus it may be wasteful to constantly execute the BEST INVESTMENT ALGORITHM. This analysis sets the basis for our forthcoming design of policies that, under partial knowledge of the parameter dynamics, allows the decision maker to schedule in advance when future actions need to be taken.

For convenience, denote by

$$\theta = (A, \alpha, \beta) \in \mathbb{Y} = \mathbb{R}_{\geq 0}^{|V|\times|V|} \times \mathbb{P}(n) \times \mathbb{R}_{\geq 0}$$

the triplet that consists of an adjacency matrix A for the graph G, a probability vector α on the set of paths $\mathcal{P}(s)$ that start at s and end at a sink, and a reward β associated with correctly preparing for a target reaching the goal. When necessary, we add a subindex to denote that an element corresponds to the parameters specified by θ. For instance, J_θ and M_θ denote the objective function (1) and the optimal stopping problem associated to θ, respectively. Finally, we denote by $u_\theta^k \in \mathcal{U}$ the kth best control policy for the problem with data θ. Therefore,

$$J_\theta(u_\theta^1) \geq J_\theta(u_\theta^2) \geq \cdots \geq J_\theta(u_\theta^{|\mathcal{U}|}). \tag{5}$$

According to this notation, $J_{\theta'}(u_\theta^k)$ is the value of the objective function (1) associated to θ' obtained by using the kth best control policy for the problem with data θ. Ideally, given the problem with data $\theta \in \mathbb{Y}$, we would like to determine the set of parameters with the same optimal control policy, i.e.,

$$\mathcal{Y}(\theta) = \{\theta' \in \mathbb{Y} \,|\, u_{\theta'}^1 = u_\theta^1\}.$$

Unfortunately, finding a general closed-form expression for $\mathcal{Y}(\theta)$ is not possible. Instead, our strategy is to find a subset of $\mathcal{Y}(\theta)$ which can be described explicitly.

We start by stating a result that bounds the changes in the value of the objective function for the kth best control policy in terms of the changes in the problem data.

Lemma VI.1 (Bounds on performance variation of a control policy under parameter changes) *For $\theta = (A, \alpha, \beta)$, $\theta' = (A', \alpha', \beta') \in \mathbb{Y}$, let*

$$\begin{aligned}\Delta^+(\theta, \theta') = \sum_{\mu=1}^{n} \max_{x \in \mathfrak{S}(p_\mu)} \{&c(x)\alpha_\mu - c'(x)\alpha'_\mu \\ &+ \alpha'_\mu \beta' \mathscr{P}'(g|\,x) - \alpha_\mu \beta \mathscr{P}(g|\,x)\},\\ \Delta^-(\theta, \theta') = \sum_{\mu=1}^{n} \min_{x \in \mathfrak{S}(p_\mu)} \{&c(x)\alpha_\mu - c'(x)\alpha'_\mu \\ &+ \alpha'_\mu \beta' \mathscr{P}'(g|\,x) - \alpha_\mu \beta \mathscr{P}(g|\,x)\}.\end{aligned}$$

Then, for any $k \in \{1, \dots, |\mathcal{U}|\}$,

$$\Delta^-(\theta, \theta') \leq J_{\theta'}(u_\theta^k) - J_\theta(u_\theta^k) \leq \Delta^+(\theta, \theta'). \tag{6}$$

Proof: This can be seen by expanding out

$$\begin{aligned}J_{\theta'}(u_\theta^k) &= \sum_{x \in \text{Inv}_{u_\theta^k}} \mathscr{P}'(x|\,s) Q_{\theta'}(x)\\ &= \sum_{\{\mu \,|\, x_\mu^k \in \text{Inv}_{u_\theta^k}\}} \alpha'_\mu(\beta' \mathscr{P}'(g|\,x) - c'(x)),\end{aligned}$$

and verifying that (6) follows. ■

Combining Lemma VI.1 with the ordering (5), we can deduce the following useful result.

Corollary VI.2 (Bounds on performance of different control policies under parameter changes) *For θ, $\theta' \in \mathbb{Y}$ and any $k \in \{1, \dots, |\mathcal{U}|\}$,*

$$J_{\theta'}(u_{\theta'}^z) \leq J_\theta(u_\theta^k) + \Delta^+(\theta, \theta'), \quad \text{for } z \geq k, \tag{7a}$$
$$J_{\theta'}(u_{\theta'}^z) \geq J_\theta(u_\theta^k) + \Delta^-(\theta, \theta'), \quad \text{for } z \leq k. \tag{7b}$$

Proof: We prove the first statement here. The proof of the second statement is analogous. Note that, for all $z \geq k$,

$$J_{\theta'}(u_\theta^z) \leq J_\theta(u_\theta^z) + \Delta^+(\theta, \theta') \leq J_\theta(u_\theta^k) + \Delta^+(\theta, \theta'),$$

where we have used Lemma VI.1 in the first inequality and the ordering (5) in the second. Therefore, the right-hand side is an upper bound on the performance of at least $|\mathcal{U}| - k + 1$ control policies. In other words, the inequality

$$J_{\theta'}(u) > J_\theta(u_\theta^k) + \Delta^+(\theta, \theta') \tag{8}$$

can only be true for at most $k - 1$ control policies $u \in \mathcal{U}$. To show that (7a) holds, we now reason by contradiction. Suppose there exists $z \geq k$ such that $J_{\theta'}(u_{\theta'}^z) > J_\theta(u_\theta^k) + \Delta^+(\theta, \theta')$. Then, because of the ordering (5), we deduce that $J_{\theta'}(u_{\theta'}^l) \geq J_{\theta'}(u_{\theta'}^z) > J_\theta(u_\theta^k) + \Delta^+(\theta, \theta')$ for all $l \in \{1, \dots, z\}$. Since $z \geq k$, this contradicts the fact that the inequality (8) can only be true for at most $k - 1$ control policies. ■

The next result builds on Lemma VI.1 to provide an easy test of whether the solution to M_θ remains optimal for $M_{\theta'}$.

Proposition VI.3 (Criterium for best solution to remain optimal) *For $\theta \in \mathbb{Y}$, let*

$$\widetilde{\mathcal{Y}}(\theta) = \{\theta' \in \mathbb{Y} \,|\, J_{\theta'}(u_\theta^1) \geq J_\theta(u_\theta^2) + \Delta^+(\theta, \theta')\}. \tag{9}$$

Then, $\widetilde{\mathcal{Y}}(\theta) \subset \mathcal{Y}(\theta)$.

Proof: To prove the result, we must show that if $\theta' \in \widetilde{\mathcal{Y}}(\theta)$, then $\theta' \in \mathcal{Y}(\theta)$, i.e., $u^1_{\theta'} = u^1_\theta$. We begin by noting that, given the ordering of values (5) associated to the control policies $u^1_\theta, u^2_\theta, \dots$, condition (9) implies that, for any $k \geq 2$,

$$J_{\theta'}(u^1_\theta) \geq J_\theta(u^k_\theta) + \Delta^+(\theta, \theta').$$

Combining this inequality with Lemma VI.1, we deduce that

$$J_{\theta'}(u^1_\theta) \geq J_{\theta'}(u^k_\theta),$$

implying that u^1_θ is better than u^k_θ, $k \geq 2$, for the problem with data θ', i.e., u^1_θ remains optimal. ∎

Proposition VI.3 provides a checkable condition to determine if the optimal control policy remains optimal after the problem parameters change. Observing (9), the role that the second best solution plays in evaluating these conditions becomes clear.

For the problem described in Figures 1-2, we run the BEST INVESTMENT ALGORITHM to compute the optimal solution u^1_θ and the SECOND BEST INVESTMENT ALGORITHM to find u^2_θ. We then randomly vary the data of the problem $\theta = (A, \alpha, \beta)$ by up to 3 percent of their previous value in subsequent iterations. At each step, instead of running these algorithms again, we can check whether the new parameters belong to $\widetilde{\mathcal{Y}}(\theta)$. If they do not, only then does this trigger a re-execution of the BEST INVESTMENT ALGORITHM. Figure 6 demonstrates the benefit of performing this additional test. In this case we see that the condition is satisfied until iteration 25 and thus the optimal solution does not need to be recomputed until then. Corollary VI.2 implies that $J_\theta(u^2_\theta) + \Delta^+(\theta, \theta')$ is an upper bound on the value obtained by any suboptimal policy $J_{\theta'}(u^k_{\theta'})$ for $k \geq 2$. Although $u^2_{\theta'}$ does not need to be recomputed at each timestep, Figure 6 shows the value $J_{\theta'}(u^2_{\theta'})$ to illustrate this upper bound on suboptimal policies.

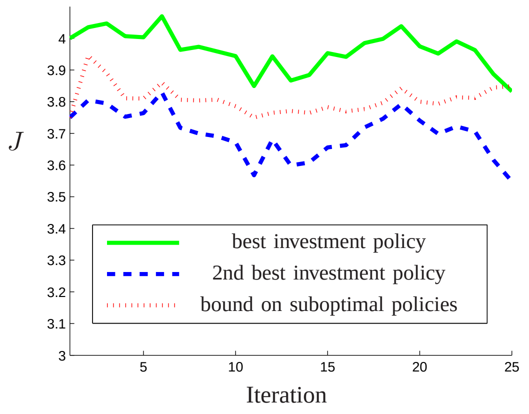


Fig. 6. Illustration of the application of Proposition VI.3 for the problem described in Figures 1-2. In each iteration, the problem parameters are randomly changed by up to 3 percent. The curves correspond to the value obtained by the optimal investment (solid), the second best investment (dashed), and the upper bound on all suboptimal investment policies (dotted).

VII. SELF-TRIGGERED ACQUISITION&DECISION ALGORITHM

Having identified in Section VI conditions under which the best solution remains optimal under parameter changes, here we turn our attention to the study of scenarios where parameters have dynamics, and the decision maker has some (possibly partial) knowledge of it. We assume the decision maker can obtain the true parameters at any point in time, but that doing so has some associated cost. The objective is then to determine how long the decision maker can go without knowing the exact values of the parameters while ensuring that its currently implemented control policy remains optimal.

A. Information available to the decision maker

Here, we describe the information available to the decision maker about the parameter dynamics. We assume that the timescale of the target motion in the network is much faster than the timescale of the evolution of the parameters. Let the parameter evolution $\{\theta(\ell) \mid \ell \in \mathbb{Z}_{\geq 0}\}$ be described by

$$\theta(\ell + 1) = \theta(\ell) + w(\ell) + \gamma(\ell). \tag{10}$$

The model that describes the decision maker's knowledge is as follows. The sequence $\{w(\ell) \mid \ell \in \mathbb{Z}_{\geq 0}\}$ is a priori known by the decision maker, whereas the sequence $\{\gamma(\ell) \mid \ell \in \mathbb{Z}_{\geq 0}\}$ is not. We assume that the magnitude of each component of γ is upper bounded. Specifically, if the components of γ are $(\gamma_{1,1}, \dots, \gamma_{|V|,|V|}, \gamma_1, \dots, \gamma_n, \gamma_\beta)$, then the decision maker is also aware of a vector $\bar{\gamma}$ such that

$$|\gamma_{i,j}(\ell)| \leq \bar{\gamma}_{i,j}, \quad |\gamma_\mu(\ell)| \leq \bar{\gamma}_\mu, \quad \text{and} \quad |\gamma_\beta(\ell)| \leq \bar{\gamma}_\beta, \tag{11}$$

for all $i, j \in \{1, \dots, |V|\}$ and $\mu \in \{1, \dots, n\}$. Therefore, at any given time $\ell \in \mathbb{Z}_{\geq 0}$, the decision maker has some uncertainty about the exact value of the parameters $\theta(\ell)$ (note that if $\bar{\gamma} = 0$, then there is no uncertainty at all). Finally, we assume that the decision maker has the ability to acquire the true values of the parameters but that this has an associated cost that it would rather not pay. In the face of this uncertainty, the objective of the decision maker is to determine for how long it can operate without exact knowledge of the parameters and still guarantee that its last computed best investment policy remains optimal. Our analysis starts by considering initial parameter values $\theta(\ell_*)$ corresponding to the last time ℓ_* for which the decision maker computed the best and second best investment policies. Note that one can rewrite (10) as

$$\theta(\ell) = \theta(\ell_*) + v(\ell) + \delta(\ell),$$

where $v(\ell) = \sum_{k=\ell_*}^{\ell-1} w(k)$ and $\delta(\ell) = \sum_{k=\ell_*}^{\ell-1} \gamma(k)$. Note also that (11) implies that δ is upper bounded linearly in time.

B. Rationale for algorithm design

To simplify the exposition, in this section we reason for general $\widehat{\theta}$ and $\widehat{\theta}' = \widehat{\theta} + v + \delta$. One can readily draw the connection with the parameter dynamics described above by setting $\widehat{\theta} = \theta(\ell_*)$ and $\widehat{\theta}' = \theta(\ell)$, for $\ell \geq \ell_*$. Given the uncertainty of $\widehat{\theta}'$, the decision maker cannot test the condition (9) directly. Instead, our approach leverages the partial knowledge $\widehat{\theta} + v$ about $\widehat{\theta}'$ by checking (9) for $\widehat{\theta}$ and $\widehat{\theta} + v$, i.e., whether

$$J_{\widehat{\theta}+v}(u^1_{\widehat{\theta}}) \geq J_{\widehat{\theta}}(u^2_{\widehat{\theta}}) + \Delta^+(\widehat{\theta}, \widehat{\theta} + v) \tag{12}$$

holds. If this condition fails, we cannot make any guarantee about the optimal solution corresponding to $\widehat{\theta}'$ and thus it is necessary for the decision maker to access the true parameters. However, if (12) holds, then $u^1_{\widehat{\theta}+v} = u^1_{\widehat{\theta}}$. To determine whether $u^1_{\widehat{\theta}'}$ is the same policy as these, we utilize (9) to check if

$$J_{\widehat{\theta}'}(u^1_{\widehat{\theta}+v}) = J_{\widehat{\theta}'}(u^1_{\widehat{\theta}}) \geq J_{\widehat{\theta}+v}(u^2_{\widehat{\theta}+v}) + \Delta^+(\widehat{\theta}+v, \widehat{\theta}') \quad (13)$$

holds. Unfortunately, since both $\widehat{\theta}'$ and $u^2_{\widehat{\theta}+v}$ are unknown, we cannot evaluate either side of (13) directly. The following result is our first step towards solving this dilemma.

Lemma VII.1 (Alternative criterium for best solution to remain optimal) *For $\widehat{\theta}$ and $\widehat{\theta}' = \widehat{\theta} + v + \delta$, if*

$$\begin{aligned} &J_{\widehat{\theta}+v}(u^1_{\widehat{\theta}}) + \Delta^-(\widehat{\theta}+v, \widehat{\theta}+v+\delta) \\ &\geq J_{\widehat{\theta}}(u^2_{\widehat{\theta}}) + \Delta^+(\widehat{\theta}, \widehat{\theta}+v) + \Delta^+(\widehat{\theta}+v, \widehat{\theta}+v+\delta), \quad (14) \end{aligned}$$

then both (12) *and* (13) *hold.*

Proof: The fact that (14) implies (12) readily follows by noting that $\Delta^+(\theta,\theta') - \Delta^-(\theta,\theta') \geq 0$ for any θ, θ'. To show (13), with the notation of Corollary VI.2, letting $\theta = \widehat{\theta}$, $\theta' = \widehat{\theta}'$, and $k = 2$, we can upper bound the RHS of (13) by

$$\begin{aligned} &J_{\widehat{\theta}+v}(u^2_{\widehat{\theta}+v}) + \Delta^+(\widehat{\theta}+v, \widehat{\theta}') \\ &\qquad \leq J_{\widehat{\theta}}(u^2_{\widehat{\theta}}) + \Delta^+(\widehat{\theta}, \widehat{\theta}+v) + \Delta^+(\widehat{\theta}+v, \widehat{\theta}'). \end{aligned}$$

On the other hand, since (12) holds, we have that $u^1_{\widehat{\theta}+v} = u^1_{\widehat{\theta}}$ by Proposition VI.3. This fact, together with Lemma VI.1, allows us to lower bound the LHS of (13) by

$$J_{\widehat{\theta}'}(u^1_{\widehat{\theta}+v}) \geq J_{\widehat{\theta}+v}(u^1_{\widehat{\theta}}) + \Delta^-(\widehat{\theta}+v, \widehat{\theta}').$$

As a consequence, we deduce that (14) implies that (13) holds. ■

Lemma VII.1 provides an alternative condition to (13) that is easier to check because it does not require knowledge of $u^2_{\widehat{\theta}}$. However, the presence of the unknown vector δ still makes it uncheckable by the decision maker. Therefore, our next step consists of using the knowledge (11) available to the decision maker to upper bound the term

$$\Delta^+(\theta,\theta') - \Delta^-(\theta,\theta'), \quad (15)$$

for $\theta = \widehat{\theta} + v$ and $\theta' = \widehat{\theta} + v + \delta$. Given the result in Lemma VI.1, we refer to (15) as the *size of performance variation* between θ and θ'. Before stating our next result, we need to introduce a piece of notation. Let $m(\theta) = \min\{\mathrm{lngth}^{sw}(x, g) \,|\, x \in X \text{ such that } g \notin x\}$ be the minimum shortest weighted length of all states that do not contain g.

Lemma VII.2 (Bounds on size of performance variation) *Given θ and θ', let the magnitude of $\theta' - \theta$ be bounded by some vector ω component-wise. Assume there exists $d_* > 0$ such that $d \mapsto f(1/d)$ is globally Lipschitz on $[d_*, \infty)$ with Lipschitz constant D, i.e., $|f\left(\frac{1}{d}\right) - f\left(\frac{1}{d'}\right)| \leq D|d - d'|$, for all d, $d' \geq d_*$. For $\theta \in \mathbb{Y}$, define*

$$G(\theta,\omega) = \omega_\beta + D\sum_{i,j=1}^{|V|} \omega_{i,j} + \sum_{\mu=1}^{n} K_\mu \omega_\mu,$$

with $K_\mu = \mathrm{diam}_{x\in\mathfrak{S}(p_\mu)}\, c(x) + 2\beta(n+1)$, for $\mu \in \{1,\dots,n\}$. If $m(\theta), m(\theta') \geq d_*$, then

$$\Delta^+(\theta,\theta') - \Delta^-(\theta,\theta') \leq G(\theta,\omega).$$

Proof: Let $\mathrm{diam}_{x\in X}(g(x)) = \max_{x\in X} g(x) - \min_{x\in X} g(x)$ for any real-valued function g. Two useful properties of the diam function are that

$$\begin{aligned} \mathrm{diam}_{x\in X}\, e g(x) &= |e|\, \mathrm{diam}_{x\in X}\, g(x), \\ \mathrm{diam}_{x\in X}[g_1(x) + g_2(x)] &\leq \mathrm{diam}_{x\in X}\, g_1(x) + \mathrm{diam}_{x\in X}\, g_2(x), \end{aligned}$$

for any $e \in \mathbb{R}$ and real-valued functions g_1 and g_2. One can write out $\Delta^+(\theta,\theta') - \Delta^-(\theta,\theta')$ as

$$\begin{aligned} &\sum_{\mu=1}^{n} \mathrm{diam}_{x\in\mathfrak{S}(p_\mu)}[c(x)(\alpha_\mu - \alpha'_\mu) + \alpha'_\mu(c(x) - c'(x)) \quad (16) \\ &\quad + \alpha'_\mu \mathscr{P}'(g|\,x)(\beta' - \beta) + \beta(\alpha'_\mu \mathscr{P}'(g|\,x) - \alpha_\mu \mathscr{P}(g|\,x))]. \end{aligned}$$

Using the two properties above, (16) can be upper bounded by

$$\begin{aligned} &\sum_{\mu=1}^{n} |\alpha_\mu - \alpha'_\mu|\, \mathrm{diam}_{x\in\mathfrak{S}(p_\mu)}\, c(x) \\ &\quad + \alpha'_\mu\, \mathrm{diam}_{x\in\mathfrak{S}(p_\mu)}(c(x) - c'(x)) \\ &\quad + \alpha'_\mu |\beta' - \beta|\, \mathrm{diam}_{x\in\mathfrak{S}(p_\mu)}\, \mathscr{P}'(g|\,x) \\ &\quad + \beta\, \mathrm{diam}_{x\in\mathfrak{S}(p_\mu)}(\alpha'_\mu \mathscr{P}'(g|\,x) - \alpha_\mu \mathscr{P}(g|\,x)). \end{aligned}$$

Given the statement of the result, we need to work on all but the first term. Using the definition of the cost c and the globally Lipschitz assumption on $d \mapsto f(\frac{1}{d})$, we upper bound

$$\alpha'_\mu\, \mathrm{diam}_{x\in\mathfrak{S}(p_\mu)}(c(x) - c'(x)) \leq \alpha'_\mu D \sum_{i,j=1}^{|V|} |a'_{i,j} - a_{i,j}|.$$

The third term is readily upper bounded using that $\mathrm{diam}_{x\in\mathfrak{S}(p_\mu)}\, \mathscr{P}(g|\,x) \leq 1$. Finally, the fourth term can be dealt with as follows. Given x, let $Z(x) = \{\nu \in \mathrm{Ind}(x) \,|\, \mathrm{last}(p_\nu) = g\}$ denote the set of indices of the paths that contain x and finish at the goal g. Note that

$$\begin{aligned} &\Big(\sum_{\nu\in\mathrm{Ind}(x)} \alpha_\nu \sum_{\upsilon\in\mathrm{Ind}(x)} \alpha'_\upsilon\Big)\Big(\alpha'_\mu \mathscr{P}'(g|\,x) - \alpha_\mu \mathscr{P}(g|\,x)\Big) \\ &\quad = \alpha'_\mu \sum_{\psi\in Z(x)} \alpha'_\psi \sum_{\nu\in\mathrm{Ind}(x)} \alpha_\nu - \alpha_\mu \sum_{\chi\in Z(x)} \alpha_\chi \sum_{\upsilon\in\mathrm{Ind}(x)} \alpha'_\upsilon \\ &\quad = \alpha'_\mu \sum_{\nu\in\mathrm{Ind}(x)} \alpha_\nu \sum_{\chi\in Z(x)} (\alpha'_\chi - \alpha_\chi) \\ &\qquad + \alpha'_\mu \sum_{\chi\in Z(x)} \alpha_\chi \sum_{\nu\in\mathrm{Ind}(x)} (\alpha_\nu - \alpha'_\nu) \\ &\qquad + \sum_{\chi\in Z(x)} \alpha_\chi \sum_{\upsilon\in\mathrm{Ind}(x)} \alpha'_\upsilon(\alpha'_\mu - \alpha_\mu). \end{aligned}$$

Denoting $W(x) = \mathrm{Ind}(x) \setminus Z(x)$, one can further simplify this expression as

$$\begin{aligned} &\alpha'_\mu \sum_{\phi\in W(x)} \alpha_\phi \sum_{\chi\in Z(x)} (\alpha'_\chi - \alpha_\chi) + \alpha'_\mu \sum_{\chi\in Z(x)} \alpha_\chi \sum_{\phi\in W(x)} (\alpha_\phi - \alpha'_\phi) \\ &\quad + \sum_{\chi\in Z(x)} \alpha_\chi \sum_{\upsilon\in\mathrm{Ind}(x)} \alpha'_\upsilon(\alpha'_\mu - \alpha_\mu). \end{aligned}$$

Given that $\mathrm{Ind}(x) = Z(x) \cup W(x)$ and $\mu \in \mathrm{Ind}(x)$,

$$\max_{x\in\mathfrak{S}(p_\mu)} [\alpha'_\mu \mathscr{P}'(g|\,x) - \alpha_\mu \mathscr{P}(g|\,x)]$$

$$\leq \alpha'_\mu \max_{x\in\mathfrak{S}(p_\mu)} \Big(\frac{\sum_{\chi\in Z(x)} |\alpha'_\chi - \alpha_\chi| + \sum_{\phi\in W(x)} |\alpha_\phi - \alpha'_\phi|}{\sum_{v\in\mathrm{Ind}(x)} \alpha'_v}\Big)$$

$$+ |\alpha'_\mu - \alpha_\mu| \leq \sum_{\nu=1}^{n} |\alpha'_\nu - \alpha_\nu| + |\alpha'_\mu - \alpha_\mu|.$$

Similarly, $-\min_{x\in\mathfrak{S}(p_\mu)}[\alpha'_\mu \mathscr{P}'(g|\,x) - \alpha_\mu \mathscr{P}(g|\,x)]$ is upper bounded by the same quantity, thus

$$\mathrm{diam}_{x\in\mathfrak{S}(p_\mu)}[\alpha'_\mu \mathscr{P}'(g|\,x) - \alpha_\mu \mathscr{P}(g|\,x)]$$
$$\leq 2(|\alpha'_\mu - \alpha_\mu| + \sum_{\nu=1}^{n} |\alpha'_\nu - \alpha_\nu|),$$

from which the result follows. ■

Using Lemma VII.2 in (14) with $\theta = \widehat{\theta} + v$, $\theta' = \widehat{\theta} + v + \delta$ and $\omega = \delta$, we deduce

$$J_{\widehat{\theta}+v}(u^1_{\widehat{\theta}}) \geq J_{\widehat{\theta}}(u^2_{\widehat{\theta}}) + \Delta^+(\widehat{\theta}, \widehat{\theta}+v) + G(\widehat{\theta}+v, \delta). \quad (17)$$

Therefore, if this condition is satisfied, Lemma VII.1 implies that (12) and (13) hold, which means $u^1_{\widehat{\theta}'} = u^1_{\widehat{\theta}}$. Fortunately, (17) can be checked by the decision maker with the information it possesses. This sets the basis for the design of self-triggered policies, which we address next.

C. SELF-TRIGGERED ACQUISITION&DECISION ALGORITHM

This section presents a strategy that builds on the conditions identified in Section VII-B to determine, with the information available to the decision maker about the parameter dynamics described in Section VII-A, the longest period of time for which the best investment policy is guaranteed to remain optimal. We refer to this strategy as the SELF-TRIGGERED ACQUISITION&DECISION ALGORITHM and present it formally in Algorithm 3. The term 'self-triggered' is meant to emphasize the fact that the decision maker determines this period of time autonomously.

The output of the SELF-TRIGGERED ACQUISITION&DECISION ALGORITHM is the number of timesteps $\Delta\ell_{\text{sleep}}$ for which the decision maker can 'sleep', i.e., starting from the time ℓ at which the strategy is executed, the current optimal solution is guaranteed to remain optimal for at least $\Delta\ell_{\text{sleep}}$ timesteps.

Proposition VII.3 (Correctness of the SELF-TRIGGERED ACQUISITION&DECISION ALGORITHM) *Under the model for parameter evolution described in Section VII-A, let $\Delta\ell_{sleep}$ and u^1 be as defined by the* SELF-TRIGGERED ACQUISITION-&DECISION ALGORITHM *executed at time $\ell \in \mathbb{Z}_{\geq 0}$. Then, the control policy u^1 is guaranteed to be optimal for timesteps $\ell, \ell+1, \ldots, \ell+\Delta\ell_{sleep}-1$.*

Proof: We show that the SELF-TRIGGERED ACQUISITION&DECISION ALGORITHM ensures that condition (17) is satisfied for $\ell, \ell+1, \ldots, \ell+\Delta\ell_{\text{sleep}}-1$ and thus u^1 remains optimal. For $\ell' \geq \ell$, let $v(\ell') = \sum_{k=\ell}^{\ell'-1} w(k)$ and $\delta(\ell') = \sum_{k=\ell}^{\ell'-1} \gamma(k)$. Steps `1:-5:` of Algorithm 3 guarantee that the control policy u^1 in memory is the optimal one for the parameters $\theta(\ell)$ at time ℓ. With the notation of Section VII-B, let $\widehat{\theta} = \theta(\ell_*)$, where ℓ_* corresponds to the last time when the BEST and SECOND BEST INVESTMENT ALGORITHMS were executed, and let $v = \theta(\ell) - \theta(\ell_*) + v(\ell')$ and $\delta = \delta(\ell')$. Step `7:` ensures that (17) is satisfied where $G(\widehat{\theta}+v, \delta)$ is replaced by $G(\widehat{\theta}+v, \bar{\gamma}(\ell'-\ell))$ using the upper bound on $\delta(\ell')$ induced by (11). ■

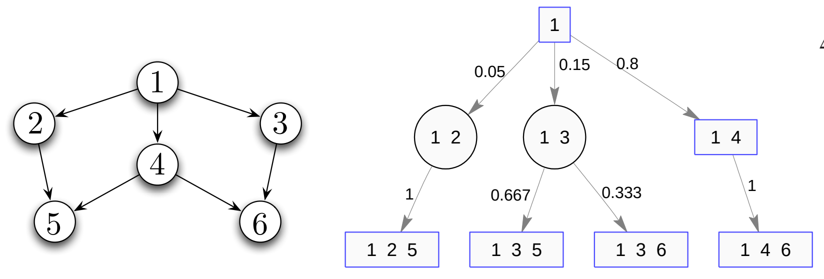
Algorithm 3 : SELF-TRIGGERED ACQUISITION&DECISION ALGORITHM

```
Information kept in memory:
 1: $\theta_{\text{old}} = \theta(\ell_*)$ {parameter vector at the last execution of the BEST and SECOND BEST INVESTMENT ALGORITHMS}
 2: $u^1 = u^1_{\theta_{\text{old}}}$ and $u^2 = u^2_{\theta_{\text{old}}}$
At current time $\ell$:
 1: acquire new parameters $\theta_{\text{new}} = \theta(\ell)$
 2: initialize $\Delta\ell_{\text{sleep}} = \infty$ and $\Delta\ell_{\text{test}} = 1$
 3: initialize $v(\ell') = \sum_{k=\ell}^{\ell'-1} w(k)$
Perform:
 1: if $J_{\theta_{\text{new}}}(u^1) < J_{\theta_{\text{new}}}(u^2) + \Delta^+(\theta_{\text{old}}, \theta_{\text{new}})$ then
 2:     execute the BEST INVESTMENT ALGORITHM and update $u^1$
 3:     execute the SECOND BEST INVESTMENT ALGORITHM and update $u^2$
 4:     set $\theta_{\text{old}} = \theta_{\text{new}}$
 5: end if
 6: while $\Delta\ell_{\text{sleep}} > \Delta\ell_{\text{test}}$ do
 7:     if $J_{\theta_{\text{new}}+v(\ell+\Delta\ell_{\text{test}})}(u^1) < J_{\theta_{\text{old}}}(u^2) + \Delta^+(\theta_{\text{old}}, \theta_{\text{new}} + v(\ell+\Delta\ell_{\text{test}})) + G(\theta_{\text{new}} + v(\ell+\Delta\ell_{\text{test}}), \bar{\gamma}\Delta\ell_{\text{test}})$ then
 8:         $\Delta\ell_{\text{sleep}} = \Delta\ell_{\text{test}}$
 9:     end if
10:     $\Delta\ell_{\text{test}} = \Delta\ell_{\text{test}} + 1$
11: end while
```

Remarkably, if the decision maker has full knowledge of how parameters evolve, i.e., $\bar{\gamma} = 0$, then the SELF-TRIGGERED ACQUISITION&DECISION ALGORITHM simply consists of checking the first time that (9) will be violated. In particular, Figure 6 can be seen as an execution of the SELF-TRIGGERED ACQUISITION&DECISION ALGORITHM for this case.

Figure 7 shows another simple example of how the SELF-TRIGGERED ACQUISITION&DECISION ALGORITHM works where three parameters are changed linearly in such a way that the initial second best control policy eventually becomes optimal. When the parameter evolution is completely unknown, the strategy yields $\Delta\ell_{\text{sleep}} = 1$. Instead, when the parameter evolution is completely known, the strategy greatly improves to $\Delta\ell_{\text{sleep}} = 19$ timesteps. This is a remarkable match with the fact that the first time the optimal solution changes is after 20 timesteps. The monotonically increasing plot in Figure 7(c) corresponds to the fact that, as the part of the parameter dynamics that is known to the decision maker becomes dominant, the periods of guaranteed optimality of the current best investment decision policy become larger.

Remark VII.4 (SELF-TRIGGERED ACQUISITION&RE-COMPUTATION ALGORITHM) An alternative, simpler version of Algorithm 3 consists of eliminating the 'if' condition in steps `1:` and `5:` so that, each time the strategy prescribes a 'wake-up call', the best and second best control policies are recomputed with the newly acquired parameters. We refer to this policy as the SELF-TRIGGERED



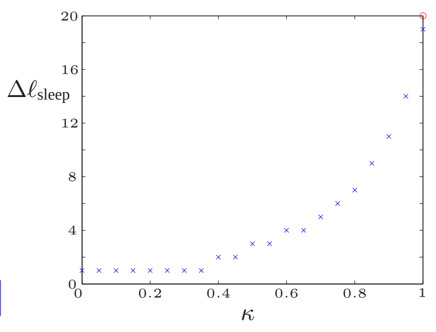


Fig. 7. Illustration of the application of the SELF-TRIGGERED ACQUISITION&DECISION ALGORITHM for the example problem displayed in (a) with $\alpha = [.05, .1, .05, .8]$. The goal of interest is Node 5. The corresponding optimal stopping problem and optimal solution (black circles) are shown in (b). In each iteration, the edge weight between Node 2 and Node 5 is decreased by 0.08, the probability α_1 is decreased by 0.002, and α_2 is increased by 0.002. The result of the SELF-TRIGGERED ACQUISITION&DECISION ALGORITHM is shown in (c) for various levels of knowledge $\kappa = \frac{|v(\ell)|}{|v(\ell)|+\bar{\gamma}}$ on the perturbations in the parameters. The perfect-knowledge case is captured by $\kappa = 1$ corresponding to $\bar{\gamma} = 0$ and recovers condition (9). The no-knowledge case is captured by $\kappa = 0$ corresponding to $v(\ell) = 0$ for all ℓ. The red circle in the top right corner of (c) corresponds to the exact time when the optimal solution changes.

ACQUISITION&RECOMPUTATION ALGORITHM. Instead, the SELF-TRIGGERED ACQUISITION&DECISION ALGORITHM aims to save on executions of the BEST and SECOND BEST INVESTMENT ALGORITHMS by checking whether the control policy u^1 in memory remains optimal for the new parameters before scheduling the next 'wake-up call'. •

D. Worst-case performance guarantees

As a byproduct of our analysis, we provide explicit guarantees on how long the optimal solution remains optimal while the problem parameters change in the worst possible way.

Corollary VII.5 (Worst-case performance guarantee) *Let the parameter evolution be described by $\theta(\ell+1) = \theta(\ell)+\gamma(\ell)$, where the magnitude of each component of γ is upper bounded by $\bar{\gamma}$ as in (11). Given $\theta(0) = \theta$, choose d_* such that $\{\theta(\ell) \mid \ell \in \mathbb{Z}_{\geq 0}\}$ satisfies $m(\theta(\ell)) \geq d_*$ for all $\ell \in \mathbb{Z}_{\geq 0}$ and $d \mapsto f(1/d)$ is globally Lipschitz on $[d_*, \infty)$ with Lipschitz constant D. Then, the number of timesteps for which the control policy u^1_θ remains optimal is lower bounded by*

$$\frac{J_\theta(u^1_\theta) - J_\theta(u^2_\theta)}{G(\theta, \bar{\gamma})}. \tag{18}$$

The proof of this result follows from the discussion in Section VII-B by using the fact that (11) induces a bound for $\delta(\ell) = \sum_{k=0}^{\ell-1} \gamma(k)$ linear in time and that G is linear in its second argument. In general, the bound provided by Corollary VII.5 is conservative because of our worst-case considerations. We consider a simple example in Figure 8 in which only one parameter is changed linearly in such a way as to decrease the performance gap between the best and second best policies. Applying the result of Corollary VII.5, we obtain 6 timesteps as a lower bound. Figure 8 shows that in fact it takes 31 iterations until the optimal solution changes. This mismatch can be traced back to the proof of Lemma VII.2 where we bound the size of performance variation.

Remark VII.6 (Connection between the robustness of the best solution and its performance gap with the second best solution) From Corollary VII.5 it is clear that the larger

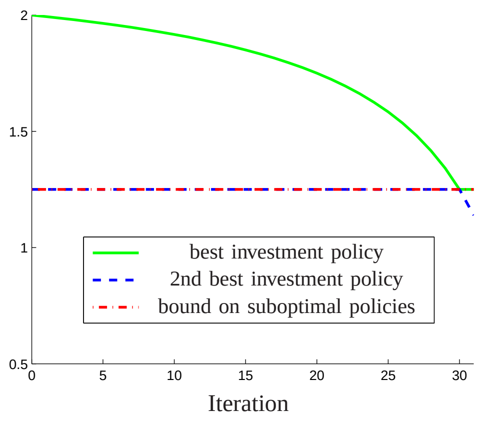


Fig. 8. Illustration of the application of Corollary VII.5 for the simple example problem displayed in Figure 7. In each iteration, the edge weight between Node 2 and Node 5 is decreased by 0.05. The curves in (c) correspond to the value obtained by the optimal investment (solid), the second best investment (dashed), and the upper bound on all suboptimal investment policies (dotted). The optimal solution changes after 31 iterations, whereas the worst-case lower bound given by Corollary VII.5 is 6.

the initial performance gap between the best and second best control policies, the more 'robust' the optimal solution is. As noted in Remark V.5, the second best control policy u^2_θ may be different for M_{inv} and $\widehat{M}_{\text{inv}}$. Since the state space $\hat{X}$ of $\widehat{M}_{\text{inv}}$ is contained in the state space X of M_{inv}, the allowable control policies for $\widehat{M}_{\text{inv}}$ are a subset of the control policies for M_{inv}. Therefore, the performance of the second best control policy of $\widehat{M}_{\text{inv}}$ can be no worse than that of the second best control policy of M_{inv}, and thus we can make better guarantees on $\widehat{M}_{\text{inv}}$. •

VIII. CONCLUSIONS

We have considered a class of problems where targets emerge from some known location and move towards some unknown destination in a weighted acyclic digraph. We have designed the BEST INVESTMENT ALGORITHM and shown that it is guaranteed to find the optimal control policy for deciding when to make preparations for the arrival of a target at a specific destination. We have also designed the SECOND BEST INVESTMENT ALGORITHM to find the second-to-optimal control policy and used it to investigate the robustness of the optimal solution against changes in the problem parameters. We have built on these conditions to obtain lower bounds, under arbitrary dynamics of the problem parameters, on the number of timesteps until the optimal solution changes. Our study has resulted in the synthesis of the SELF-TRIGGERED ACQUISITION&DECISION ALGORITHM to schedule in advance when future actions should be taken. Future work will be devoted to studying the setup where the decision maker only has access to some nodes of the network of roads or the sensors are noisy and may therefore have an incomplete knowledge of target histories; the case where the parameters of the problem are changing quickly as compared to how fast the targets move through the network; and understanding how the parameters of the problem must be selected in order to make optimal an a priori chosen investment policy.

ACKNOWLEDGMENTS

This research was partially supported by NSF award CCF-0917166.

REFERENCES

[1] N. V. Sahinidis, “Optimization under uncertainty: State-of-the-art and opportunities,” *Computers and Chemical Engineering*, vol. 28, pp. 971–983, 2004.

[2] D. P. Bertsekas, *Dynamic Programming and Optimal Control, Vol. 1*. Athena Scientific, 2 ed., 2001.

[3] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics, New York: Wiley, 2008.

[4] J. R. Birge and F. Louveaux, *Introduction to Stochastic Programming*. Springer Series in Operations Research, New York: Springer, 1997.

[5] Y. Kadota, M. Kurano, and M. Yasuda, “Utility-optimal stopping in a denumerable Markov chain,” *Bulletin of informatics and cybernetics*, vol. 28, no. 1, pp. 15–21, 1996.

[6] A. N. Shiryaev, *Optimal Stopping Rules*. Springer, 1978.

[7] T. S. Ferguson, *Optimal Stopping and Applications*. University of California, Los Angeles, 2008.

[8] N. H. Bingham and G. Peskir, “Optimal stopping and dynamic programming,” in *Encyclopedia of Quantitative Risk Analysis and Assessment* (E. L. Melnick and B. Everitt, eds.), vol. 1, pp. 1236–1243, Chichester, England: Wiley, 2008.

[9] A. Ruiz-Moncayo, “Optimal stopping for functions of Markov chains,” *The Annals of Mathematical Statistics*, vol. 39, no. 6, pp. 1905–1912, 1968.

[10] H. J. Kushner, “Computational procedures for optimal stopping problems for Markov chains,” *Journal of Mathematical Analysis and Applications*, vol. 25, no. 3, pp. 607–615, 1969.

[11] I. Sonin, “The elimination algorithm and its application to the optimal stopping problem,” in *IEEE Conf. on Decision and Control*, (San Diego, CA), Dec. 1997.

[12] I. Sonin, “The optimal stopping of Markov chain and recursive solution of Poisson and Bellman equations,” in *The Shiryaev Festschrift: From Stochastic Calculus to Mathematical Finance* (Y. Kabanov, R. Lipster, and J. Stoyanov, eds.), vol. XXXVIII, pp. 609–621, New York: Springer, 2006.

[13] M. Huang and G. N. Nair, “Detection of random targets in sensor networks with applications,” in *IFAC World Congress*, (Prague, CZ), July 2005. Electronic proceedings.

[14] G. E. Monahan, “Optimal stopping in a partially observable binary-valued Markov chain with costly perfect information,” *Journal of Applied Probability*, vol. 19, no. 1, pp. 72–81, 1982.

[15] M. L. Liu and N. V. Sahindis, “Optimization in process planning under uncertainty,” *Industrial & Engineering Chemistry Research*, vol. 35, no. 11, pp. 4154–4165, 1996.

[16] A. Bemporad, D. M. de la Peña, and P. Piazzesi, “Optimal control of investments for quality of supply improvement in electrical energy distribution networks,” *Automatica*, vol. 42, no. 8, pp. 1331–1336, 2006.

[17] K. D. Glazebrook, P. S. Ansell, R. T. Dunn, and R. R. Lumley, “On the optimal allocation of service to impatient tasks,” *Journal of Applied Probability*, vol. 41, no. 1, pp. 51–72, 2004.

[18] A. Thiele, “Robust stochastic programming with uncertain probabilities,” *IMA Journal of Management Mathematics*, vol. 19, pp. 289–321, 2008.

[19] M. Velasco, P. Marti, and J. M. Fuertes, “The self triggered task model for real-time control systems,” in *Proceedings of the 24th IEEE Real-Time Systems Symposium*, pp. 67–70, 2003.

[20] X. Wang and M. D. Lemmon, “Self-triggered feedback control systems with finite-gain L_2 stability,” *IEEE Transactions on Automatic Control*, vol. 54, no. 3, pp. 452–467, 2009.

[21] A. Anta and P. Tabuada, “To sample or not to sample: self-triggered control for nonlinear systems,” *IEEE Transactions on Automatic Control*, vol. 55, no. 9, pp. 2030–2042, 2010.

[22] C. Nowzari and J. Cortés, “Self-triggered coordination of robotic networks for optimal deployment,” *Automatica*, vol. 48, no. 6, pp. 1077–1087, 2012.