Robust optimal decision policies for servicing targets in acyclic digraphs

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. We show how this problem can be formulated as an optimal stopping problem on a Markov chain, which sets the basis for the introduction of the BEST INVESTMENT ALGORITHM. Our strategy prescribes when investments must be made conditioned on the target's motion along the digraph. We establish the optimality of this policy and examine its robustness against changing conditions of the problem which allows us to identify a sufficient condition that determines whether the solution computed by the BEST INVESTMENT ALGORITHM remains optimal under changes in the problem data. Several simulations illustrate our results.**

I. INTRODUCTION

This paper studies a decision problem in which targets appear at a known location and move through a weighted acyclic digraph to some unknown destinations. The graph is an abstraction to represent connections available to the targets between points in an environment. Sensors deployed over the nodes of the graph report the presence of a target to a decision maker which identifies the target's potential destination and prepares accordingly (for instance, by committing some resources to the destination). Our model imposes a timing trade-off on the decision maker: 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.

Literature review: Optimization problems under uncertainty appear in a plethora of different scenarios including supply management, resource allocation, queuing, and servicing problems. A typical example of this is found in [1], where an optimal control must be found given stochastic observations; however, the algorithms are usually not scalable with the size of the problem. To this point, some works such as [2] are dedicated to studying heuristic approaches to find good suboptimal control policies with reduced computation times. A discussion of current techniques and challenges related to these problems is documented in [3]. The problem we pose can be cast as an optimal stopping problem on a directed tree for which we are able to find a scalable algorithmic solution. A broad exposition of optimal stopping problems and their applications are presented in [4], [5], [6]. A general discrete time and space optimal stopping problem is studied in [7], where an elimination algorithm is proposed to speed up the computation of the optimal solution. Versions of this problem in which there are not only uncertainties in the dynamics, but also in the observations, have also been studied [8]. These problems are often solved using dynamic and stochastic programming techniques [9], [10], [11]. Another related area includes robust Markov decision problems, in which the probability distributions themselves are uncertain, e.g. [12]. We point out here that the notion of robustness considered here is different from conventional definitions. Normally, robustness studies conditions for which an algorithm can still find an 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 are interested in how robust it is to changing problem parameters. For reasons of space, all proofs are omitted and will appear elsewhere.

Statement of contributions: We start by formalizing the decision problem described above which corresponds to the optimization 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 and establish the equivalence of finding the optimal control policy with that of finding the optimal stopping set. Second, we exploit this duality to design algorithms that find the optimal and second-to-optimal investment policies and characterize their correctness and time complexities. The BEST INVESTMENT ALGORITHM is a dynamic programming-based strategy that feeds its solution to the SECOND BEST INVESTMENT ALGORITHM to find the best and second-to-best control policies. 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. Several simulations illustrate our results.

II. PRELIMINARIES

Let $\mathbb{R}$, $\mathbb{R}_{\geq 0}$, $\mathbb{N}$ be the set of real, nonnegative real, and natural numbers, respectively. The cardinality of a set is denoted $|\cdot|$.

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

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

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

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 there exists a path 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., $\text{last}(p) = v_m$. The length and weighted length of p are $\text{lngth}(p) = |p| - 1$ and $\text{lngth}^w(p) = \sum_{k=1}^{m-1} a_{i_k i_{k+1}}$, respectively. Given a sink g and a path p, let $\text{lngth}^{sw}(p, g)$ denote the weighted length of the shortest path from $\text{last}(p)$ to g,

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

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 v^* such that there exists a unique path from the root to each vertex. A useful property of a rooted tree is that every vertex that is not 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 $\{x_k \,|\, k \in \mathbb{N}\}$, 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 $Q : X \to \mathbb{R}$. 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 $\{x_k \,|\, k \in \mathbb{N}\}$ specified by M and x_0. The goal of the problem is to find a set of halting states $Y \subset X$ that maximizes the function $E_{x_0}[Q(x_\tau)]$, where x_τ is the first time the Markov chain enters Y. A maximizer of this function is an *optimal stopping set* $Y^* \subset X$. Defining the *value function* $\mathcal{V}^*(x_0) = \max_{k \in \mathbb{N}} E_{x_0}[Q(x_k)]$, optimal stopping sets can alternatively be defined by

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

III. Problem statement

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 corresponds to the cost it takes to traverse it (e.g., larger weights correspond to longer times or higher energy cost). Sensors that are able to detect the presence of a target are deployed over the graph nodes and transmit information to a decision maker. The decision maker must decide whether or not to prepare for the arrival of the target at a goal g 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 a goal in anticipation of a target's arrival. Our model specifies that the longer it takes the target to arrive, the less costly it is to make an investment for that goal because there is more time available to prepare; however, if an investment is made and the target does not arrive, the investment is wasted. For instance, if resources must be sent to the goal, it will be easier to do with more time available. Once a decision to invest has been made, it cannot be retracted.

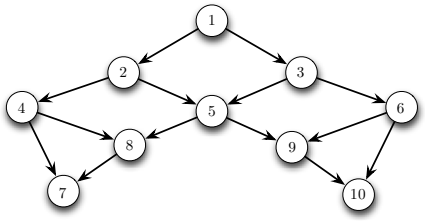

Fig. 1. Example network of roads modeled as a weighted acyclic digraph. All edge weights are equal to 1. There are $|\mathcal{P}(s)| = 8$ paths starting at the source (node 1) and ending at a sink (either node 7 or 10). The probabilities associated to these paths are given by the vector $\alpha^s = [.05,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

A. Probabilistic model for target motion

The path chosen by target T along the digraph G is unknown to the decision maker, who instead uses the following probabilistic model. Let $p_T \in \mathcal{P}(s)$ denote the path of T. The set of trajectories (or histories) that can be observed by the decision maker as T moves is $\mathfrak{S}(p_T)$. The set of all possibly observable trajectories is given by $\mathcal{H}(G) = \bigcup_{p \in \mathcal{P}(s)} \mathfrak{S}(p)$.

Let $n = |\mathcal{P}(s)|$ and assign labels $\{1, \dots, n\}$ to the paths in $\mathcal{P}(s)$. Let α^s be a probability vector, where α_i^s is the probability that target T takes path i. 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 for the targets.

Using this model, the decision maker can infer a target's future actions as it moves through the digraph. Given history $h \in \mathcal{H}$, let $\text{Ind}(h) = \{i \in \{1, \dots, n\} \,|\, h \in \mathfrak{S}(p_i)\}$ denote the set of indices that the target could possibly be on, or are *indistinguishable* to the decision maker. Then, the decision maker can compute the probability that $p_T = p_i$ as

$$\mathscr{P}(p_T = p_i | \, h) = \begin{cases} \frac{\alpha_i^s}{\sum_{j \in \text{Ind}(h)} \alpha_j^s}, & \text{if } i \in \text{Ind}(h), \\ 0, & \text{otherwise.} \end{cases}$$

The decision maker can also compute the probability that the target will eventually go to a vertex $v \in V$,

$$\mathscr{P}(v|\, h) = \sum_{\{i \in \{1,\dots,n\} \,|\, v \in p_i\}} \mathscr{P}(p_T = p_i|\, h).$$

This evaluates to 1 if v is in h and 0 if v is not in $\mathcal{R}(\text{last}(h))$.

B. Allowable control policies

As targets move through the digraph, the decision maker decides when an investment should be made for each goal, if ever. For simplicity of presentation and without loss of generality, the paper considers investment decisions for one specific goal g (for multiple goals, our policy can be applied to each one of them). We will suppress the dependence on g when it is clear. A control strategy is 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 $g \in \mathcal{S}$. Throughout the paper we consider control strategies that prescribe at most one investment along any given path because an investment only needs to be made one time. 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 $h_u(p) = \emptyset$. The set of all possible investment histories for u is $\text{Inv}_u = \cup_{i \in \{1,\dots,n\}} \{h_u(p_i)\}$.

C. Objective function

We must now define the objective function for the decision maker to optimize. We first present a model for the cost of investing in goal g for a target with history $h \in \mathcal{H}$ as

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

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

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

Since the target's path is unknown a priori, this reward is unknown when the investment is made at $h_u(p_T) \in \text{Inv}_u$. Thus, we must instead define an expected reward as

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

The decision maker seeks to maximize $E_{h_u(p_T)}[R_u(p_T)] - c(h_u(p_T))$. 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,

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

IV. Optimal stopping problem formulation

In this section we introduce an optimal stopping problem and establish its equivalence to the investment decision problem.

A. Optimal stopping problem

According to the motion model 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. Note that X is a rooted tree with the source s as the 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 $|X| \leq 1 + \sum_{p \in \mathcal{P}(s)} \text{lngth}(p)$, where the summand 1 corresponds to the trivial history s. In the worst case (if A is strictly upper triangular containing only nonzero elements), this size can get as large as $|X| = 2^{|V|-1}$.

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

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

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

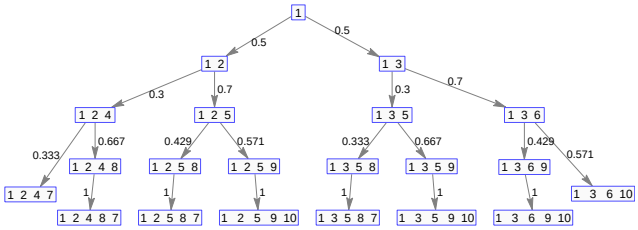

Fig. 2. State space X and transition matrix P of the optimal stopping problem associated to the problem in Figure 1. Each node represents a history $h \in \mathcal{H}$. Note that all the sinks of this tree correspond to a sink of the original digraph.

Lastly, we define the reward function as $Q(x) = \beta \mathscr{P}(g|\, x) - c(x)$, where the first and second terms correspond to the expected reward obtained from investing in goal g at state $x \in X$ and the cost of making this investment, respectively. With the problem $M_{\text{inv}} = (X, P, Q)$ defined, the next result follows from [5, Chapter 3] and the fact that X is finite.

Lemma IV.1 *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 M_{inv} with the decision problem described in Section III. To do so, we need a mapping that relates a halting state Y for the optimal stopping problem to a control policy u for the decision problem and vice versa. To this point, we define the *reduced* halting subset of a set Y for a given

initial condition x_0 as the set of all the states in Y that can be reached first by a Markov chain starting from x_0,

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

In other words, the Markov chain cannot reach states in $Y \setminus Y_{x_0}$ without passing through a state in Y_{x_0}. Interestingly, $E_{x_0}[Q(x_\tau)] = E_{x_0}[Q(x_{\tau'})]$, 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 associate the control policy

$$u_Y(x) = \begin{cases} \texttt{invest}, & \text{if } x \in Y_s, \\ \texttt{not-invest}, & \text{otherwise.} \end{cases} \tag{2}$$

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 to the problem posed in Section III.

Proposition IV.2 *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.*

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^*, cf. Section II-B. Before proceeding with the design of algorithms to accomplish this, it is advantageous to reduce the state space of M_{inv}, since this naturally results in lower algorithmic complexities.

The approach is reminiscent of techniques like the Elimination Algorithm [13] where states that trivially do not belong to the optimal set are eliminated. We start by defining a *cluster* $C = (x_1, \ldots, x_m)$ as a maximal path through the state space X such that $\mathcal{N}^{\text{out}}(x_k) = \{x_{k+1}\}$ for $k \in \{1, \ldots, m-1\}$. 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. We can now state a result to reduce the problem M_{inv}.

Lemma IV.3 *Consider the problem $M_{\text{inv}} = (X, P, Q)$. Let $C^1, \ldots, C^q$ denote all q clusters in X. Then,*

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

As a consequence of Lemma IV.3, we define a new optimal stopping problem $\widehat{M}_{\text{inv}} = (\widehat{X}, \widehat{P}, \widehat{Q})$ with state space

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

The transition matrix $\widehat{P}$ is constructed by removing all rows and columns in P corresponding to the states removed from X and replacing the rows corresponding to x_1^j with the row corresponding to x_m^j for all $j \in \{1, \ldots, q\}$. Finally, the reward function $\widehat{Q}$ is just the restriction of Q to $\widehat{X}$. Lemma IV.3 guarantees that the optimal solutions on M_{inv} and $\widehat{M}_{\text{inv}}$ are the same. Note that both problems share the objective function J as it measures the performance of a control policy on the equivalent decision problem. Interestingly, one can show that $\widehat{X} = \mathcal{H}(\widehat{G})$ is the space of histories corresponding to the collapsed digraph $\widehat{G}$.

V. Optimal investment decision policies

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

A. The BEST INVESTMENT ALGORITHM

Let us consider the optimal stopping problem $M = (X, P, Q)$ with initial condition $x_0 = s$ and goal of interest g. We will find the optimal policy to the decision problem of Section III by finding the optimal stopping set to M.

Our algorithm, making use of Bellman's principle of optimality [9], begins by solving sub-problems of type M but with different initial conditions $x_0' \in X$ where the optimal solution can be easily computed. The sub-problem can be solved for x_0' once the sub-problems have been solved for all $x \in \mathcal{R}(x_0')$. This simplifies the problem for a chosen x_0' to simply deciding whether it is optimal to stop or wait at this state. We now describe the algorithm informally.

> *[Informal description]:* Choose any node $x_0' \in X$ such that the problem is unsolved for x_0' but solved for all of its descendants. Compute the value obtained if the chain is stopped at x_0' and compare it to the expected value obtained by waiting one timestep. Save the best decision, store the value, and mark x_0' as solved. Proceed iteratively until the problem is solved for the initial condition.

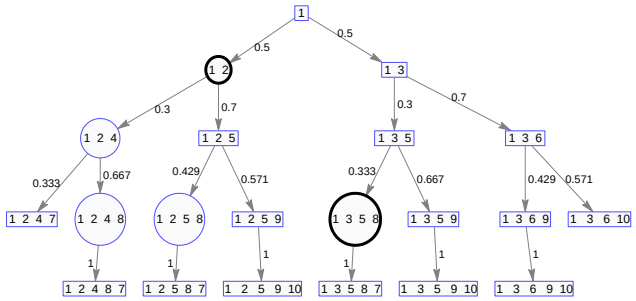

Fig. 3. Optimal solution to the problem described in Figure 1 for the goal g at Node 7, with $\beta = 20$, and cost function $f(z) = 10z$. The optimal stopping set Y^* is depicted by the 5 circular nodes and the set Y_s^* giving rise to the control policy u^* corresponds to the bold circles.

The BEST INVESTMENT ALGORITHM is presented formally in Algorithm 1. The output of Algorithm 1 is the control policy u^*, where $u^*(x) = \texttt{invest}$ for all $x \in Y_s^*$ and $u^*(x) = \texttt{not-invest}$ otherwise. Figure 3 shows the result of an execution of BEST INVESTMENT ALGORITHM.

Algorithm 1: BEST INVESTMENT ALGORITHM

Initialization:

set $\mathcal{V}^*(x) = 0$ for all $x \in X$
set $S = \{x \in X \mid \text{last}(x) \in \mathcal{S}\}$
set $Y^* = \emptyset$

Perform:

while there exists $x \notin S$ such that $y \in S$ for all $y \in \mathcal{R}(x)$ **do**
  if $Q(x) \geq \sum_{y \in \mathcal{N}^{\text{out}}(x)} \mathcal{V}^*(y) P_{x,y}$ **then**
    add x to Y^*
    set $\mathcal{V}^*(x) = Q(x)$
  else
    set $\mathcal{V}^*(x) = \sum_{y \in \mathcal{N}^{\text{out}}(x)} \mathcal{V}^*(y) P_{x,y}$
  end if
  add x to S
end while
compute $u^* = u_{Y^*}$

Algorithm 2: SECOND BEST INVESTMENT ALGORITHM

Initialization

set $\mathcal{V}_x^{\mathcal{C}} = 0$ for all $x \in X$
execute the BEST INVESTMENT ALGORITHM
compute Y_s^* from Y^*

Perform:

for $x \in Y_s^*$ **do**
  set $\mathcal{V}_x^{\mathcal{C}} = \mathcal{V}^*(s) - \mathscr{P}(x|\, s)[\mathcal{V}^*(x) - \sum_{y \in \mathcal{N}^{\text{out}}(x)} [\mathcal{V}^*(y) P_{x,y}]]$
  while $\mathcal{N}^{\text{in}}(x) \neq \emptyset$ **do**
    set $y = \mathcal{N}^{\text{in}}(x)$
    set $\mathcal{V}_y^c = \mathcal{V}^*(s) - \mathscr{P}(y|\, s)\,[\mathcal{V}^*(y) - Q(y)]$
    set $x = y$
  end while
end for
compute $\bar{x} = \text{argmax}_{x \in X}\, \mathcal{V}_x^c$
set $u' = u_{\bar{x}}^{\mathcal{C}}$

Proposition V.1 *Given the optimal stopping problem $M = (X, P, Q)$ for goal g with initial condition $x_0 = s$, the* BEST INVESTMENT ALGORITHM *finds the optimal stopping set and control policy u^* with time complexity $O(|X|)$.*

B. The SECOND BEST INVESTMENT ALGORITHM

Here we make use of the BEST INVESTMENT ALGORITHM to find the second best solution. Given an optimal stopping set Y^*, we create *candidate stopping sets*

$$\mathcal{C}_x(Y^*) = \begin{cases} Y^* \setminus \{x\}, & \text{if } x \in Y_{x_0}^*, \\ (Y^* \cup \{x\}) \setminus \mathcal{R}^{-1}(x), & \text{otherwise.} \end{cases}$$

These sets are constructed such that $u_{\mathcal{C}_x(Y^*)}(x) \neq u_{Y^*}(x)$; recall equation (2) to relate a stopping set to a control policy. For simplicity, let $u_x^{\mathcal{C}} = u_{\mathcal{C}_x(Y^*)}$. The set of control policies that we search over is then given by $\mathcal{U}^{\mathcal{C}} = \cup_{x \in X} \{u_x^{\mathcal{C}}\}$. We let $\mathcal{V}_x^{\mathcal{C}} = J(u_x^{\mathcal{C}})$ be the *candidate value* which corresponds to the value of the objective function (1) using the appropriate control policy. We now describe the algorithm informally.

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

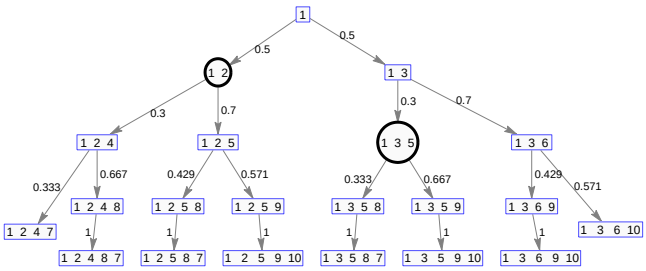

Fig. 4. Second best solution to the problem described in Figure 1 for the goal g at Node 7, with $\beta = 20$, and cost function $f(z) = 10z$. The set of investment states $\text{Inv}_{u'}$ for control policy u' is depicted by the two circular nodes. The values of the objective function for the optimal and second best control policies are given by $J(u^*) = 4.0$ and $J(u') = 3.75$, respectively.

The SECOND BEST INVESTMENT ALGORITHM is formally presented in Algorithm 2. The output is the second best control policy u'. An example execution is shown in Figure 4.

Proposition V.2 *Given the optimal stopping problem $M = (X, P, Q)$ for goal g with initial condition $x_0 = s$, the* SECOND BEST INVESTMENT ALGORITHM *finds the second best stopping set and its corresponding control policy u' with time complexity $O(|X|)$, i.e., for all $u \neq u' \neq u^*$,*

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

Remark V.3 Interestingly, even though the optimal solutions on M_{inv} and $\widehat{M}_{\text{inv}}$ are the same (cf. Lemma IV.3), 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 $\widehat{M}_{\text{inv}}$. As we show in Remark VI.3, this fact has positive implications on our analysis of the robustness of solutions.•

VI. ROBUSTNESS OF THE BEST INVESTMENT DECISION

Here we are interested in determining conditions for which the optimal solution remains optimal under changes to the problem parameters; specifically, the edge weights of the digraph, the probability model for target motion, and the reward associated with goal g. This will be beneficial because if one can easily determine that the control policy stays optimal, there is no need to re-execute the BEST INVESTMENT ALGORITHM, yielding computational savings.

For convenience, denote by $\theta = (A, \alpha, \beta) \in \mathbb{Y}$ the triplet that consists of an adjacency matrix A for G, a probability vector α on the set of paths $\mathcal{P}(s)$, and a reward β associated with correctly preparing for a target reaching g. Since the objective function now depends on both the control policy and the parameters θ, we let J_θ denote the objective function (1) associated with θ. Finally, we denote by u_θ^k the kth best control policy for the problem with data θ. Therefore,

$$J_\theta(u_\theta^1) \geq J_\theta(u_\theta^2) \geq \dots \tag{3}$$

Accordingly, $J_{\theta'}(u_\theta^k)$ 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, $\mathcal{Y}(\theta) = \{\theta' \in \mathbb{Y} \,|\, u_{\theta'}^1 = u_\theta^1\}$. However, finding a general closed-form expression for $\mathcal{Y}(\theta)$ is not possible. Instead, we

describe a subset of $\mathcal{Y}(\theta)$ by first bounding the changes in the value of the objective function for any control policy.

Lemma VI.1 *For $\theta = (A, \alpha, \beta)$ and $\theta' = (A', \alpha', \beta')$, let*

$$\Delta^u(\theta, \theta') = \sum_{i=1}^{n} \max_{x \in \mathfrak{S}(p_i)} \{c(x)\alpha_i - c'(x)\alpha'_i + \alpha'_i\beta'\mathscr{P}'(g|\,x) - \alpha_i\beta\mathscr{P}(g|\,x)\}.$$

Then, for any $k \geq 1$,

$$J_{\theta'}(u_\theta^k) - J_\theta(u_\theta^k) \leq \Delta^u(\theta, \theta'). \tag{4}$$

Proposition VI.2 *For $\theta \in \mathbb{Y}$, let*

$$\mathcal{Y}^+(\theta) = \{\theta' \in \mathbb{Y} \,|\, J_{\theta'}(u_\theta^1) \geq J_\theta(u_\theta^2) + \Delta^u(\theta, \theta')\}, \tag{5}$$

then $\mathcal{Y}^+(\theta) \subset \mathcal{Y}(\theta)$.

Proposition VI.2 provides a checkable condition to determine if the optimal control policy remains optimal under parameters changes. Observing (5), the role that the second best solution plays in evaluating this condition becomes clear.

Remark VI.3 The larger the gap between the best and second best policies is, the larger the set of parameters $\mathcal{Y}^+(\theta)$ for which the optimal policy is guaranteed to remain the same becomes. Although the second best control policy u_θ^2 may be different for M_{inv} and $\widehat{M}_{\text{inv}}$, since the state space $\widehat{X}$ 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 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 policy of M_{inv}, yielding better robustness guarantees on $\widehat{M}_{\text{inv}}$. •

For the problem described in Figures 1-4, we run the BEST INVESTMENT ALGORITHM to compute the optimal solution u_θ^1 and the SECOND BEST INVESTMENT ALGORITHM to find u_θ^2. 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 we can easily check whether the new parameters belong to $\mathcal{Y}^+(\theta)$. As long as they do, the optimal solution does not need to be recomputed. Figure 5 demonstrates the benefit of performing this check. In this case $\theta' \in \mathcal{Y}^+(\theta)$ until iteration 25 and thus the optimal solution does not need to be recomputed until then. Proposition VI.2 suggests that $J_\theta(u_\theta^2) + \Delta^u(\theta, \theta')$ is an upper bound on the value obtained by any suboptimal policy $J_{\theta'}(u_{\theta'}^k)$ for $k \geq 2$. Although $u_{\theta'}^2$ is not recomputed at each timestep, Figure 5 shows $J_{\theta'}(u_{\theta'}^2)$ to illustrate this bound.

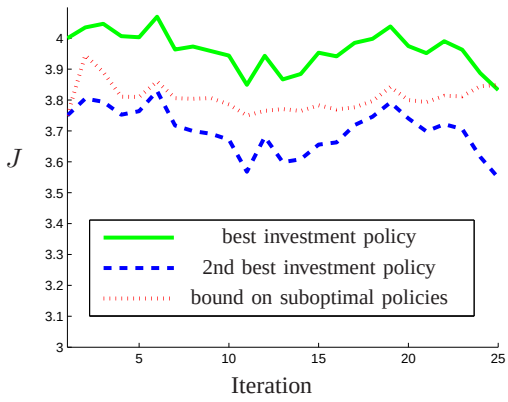

Fig. 5. Illustration of Proposition VI.2 applied to the example problem. In each iteration, the problem parameters are randomly varied. The curves correspond to the values of the optimal investment (solid), the second best investment (dashed), and the upper bound on suboptimal policies (dotted).

VII. CONCLUSIONS

We have considered a class of decision problems where targets emerge from some known location and move towards unknown destinations in a weighted acyclic digraph. We have designed the BEST INVESTMENT ALGORITHM to find the optimal control policy to the investment decision problem. We have also designed the SECOND BEST INVESTMENT ALGORITHM to find the second-to-optimal control policy and used it to synthesize a sufficient condition to see whether the solution computed before the parameters changed remains optimal. Future work will be devoted to understanding how the parameters of the problem must be selected to make optimal an a priori chosen investment policy, studying scenarios where the decision maker does not have perfect target location information, and extensions to cyclic digraphs.

ACKNOWLEDGMENTS

This research was supported by NSF award CCF-0917166.

REFERENCES

[1] D. Bernardinim, D. M. de la Peña, A. Bemporad, and E. Frazzoli, "Simultaneous optimal control and discrete stochastic sensor selection," in *Hybrid Systems: Computation and Control* (R. Majumdar and P. Tabuada, eds.), vol. 5469 of *Lecture Notes in Computer Science*, pp. 61–75, Springer Berlin, 2009.

[2] 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.

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

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

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

[6] 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.

[7] 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.

[8] 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.

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

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

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

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

[13] 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.