Connection of Optimal Stopping Time to s-t Cut Problems on Trees

Yijin Wang, Melkior Ornik, *Senior Member, IEEE*, and Roy Dong

***Abstract*—We present a method to transform any optimal stopping time problem with an underlying tree structure into an s-t min-cut problem on the same tree but with modified capacities, the details of which are lacking in existing optimal stopping time research. We also show that any s-t min/max-cut problem on a tree has an equivalent optimal stopping problem formulation. We provide a dynamic programming algorithm to solve this problem and also perform a sensitivity analysis on it. Our results imply that the s-t max-cut problem on a tree can be solved using an algorithm with runtime that is linear in the tree size.**

***Index Terms*—Optimization algorithms, optimization, stochastic optimal control.**

I. Introduction

OPTIMAL stopping has seen many applications in the setting of pricing American stock options. However, exact solutions of these problems using Dynamic Programming (DP) result in runtimes which are exponential in the time-horizon, leading instead to efforts on developing approximate solutions; see, for example, [1], [2], [3], [4].

We now briefly introduce terminology regarding some graph problems. The s-t max-flow problem is a network flow problem with the goal of sending the maximal flow from a node s to a node t in a graph. The s-t min/max-cut problems consist of partitioning a graph's nodes into two sets such that the weights of the edges across the sets is minimized/maximized. We provide more detailed definitions in Section II-A.

Chen and Goldberg [4] proposed the equivalence of the s-t max-flow/min-cut problem on trees to the minimization optimal stopping time problem $\inf_\tau E[c_\tau]$, where c_t is some non-negative random process. They note that a maximization problem can be converted to the minimization problem via

$$\sup_\tau E[c_\tau] = \bar{c} - \inf_\tau E\left[c'_\tau\right], \quad c'_t = \bar{c} - c_t \geq 0,$$

where $\bar{c}$ is a uniform upper-bound on c_t. Although this is a valid transformation of the optimization problem, a similar straightforward transformation of the capacities of the graph of an s-t max-flow/min-cut problem does not result in an equivalent s-t max-flow/min-cut solution. In this letter, we introduce a method to convert maximization or minimization optimal stopping time problems with objective functions of any sign into a min-cut problem via a $O(|V|)$ procedure, where $|V|$ is the number of nodes in the underlying tree.

Chen and Goldberg [4] also remarked that while max-flow on trees can be solved via a fast DP algorithm, from the optimal stopping time perspective, the runtime would generally be exponential in the time horizon. Nevertheless, we exploit the equivalence between the optimal stopping time and s-t min-cut problems to show some useful results. We also provide a DP algorithm and perform a sensitivity analysis on its solution in response to incremental changes in the tree capacities.

Furthermore, we show that any s-t min/max-cut problem on a tree can be transformed into an optimal stopping problem. This then provides a method for transforming an s-t max-cut problem on a tree into an equivalent s-t min-cut problem on a modified version of the tree. This means that s-t max-cut on trees can be solved exactly using a greedy algorithm. The s-t max-cut problem on general graphs is an NP-hard problem, although there are approximation algorithms; see [5], [6].

II. Notation and Terminology

We denote $\mathbf{1}_n$ to be the vector in $\mathbb{R}^n$ of ones. For a set A, 2^A denotes the power set of A. For a collection of sets $\mathcal{C}$, we let $\sigma(\mathcal{C})$ denote the σ-algebra generated by $\mathcal{C}$.

A. s-t Network Flow and Cut Problems

For a weighted graph, in which each edge has a non-negative capacity, a maximum *s-t* flow is the maximum flow from a "source" node s to a "sink" node t such that the flow through each edge does not exceed the edge's capacity. The min *s-t* cut is a partition of the graph's nodes into two sets S and T such that $s \in S$, $t \in T$, and the sum of the capacities of edges from S to T, i.e., the "cut", is minimal. The Max-flow Min-cut Duality Theorem states that the max s-t flow is equal to the min s-t cut; see, for example, [7].

In this letter, we refer to s-t max-flow and min/max-cut problem on trees with a designated root. We trivally convert a tree $\mathcal{T}$ rooted at s into a tree with a sink t by connecting

Manuscript received 16 September 2023; revised 12 November 2023; accepted 7 December 2023. Date of publication 13 December 2023; date of current version 28 December 2023. Recommended by Senior Editor K. Savla. *(Corresponding author: Yijin Wang.)*

Yijin Wang is with the Electrical and Computer Engineering Department, University of Illinois Urbana–Champaign, Urbana, IL 61801 USA (e-mail: yijinw3@illinois.edu).

Melkior Ornik is with the Aerospace Engineering Department, University of Illinois Urbana–Champaign, Urbana, IL 61801 USA (e-mail: mornik@illinois.edu).

Roy Dong is with the Industrial Engineering Department, University of Illinois Urbana–Champaign, Urbana, IL 61801 USA (e-mail: roydong@illinois.edu).

Digital Object Identifier 10.1109/LCSYS.2023.3342562

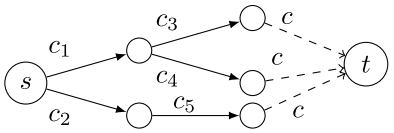

Fig. 1. Converting a tree rooted at s into one with sink t, by adding edges from the leaves to t (dashed). The labels are the edge capacities. For s-t min-cut and max-cut problems, $c = \infty$ and $-\infty$, respectively.

all leaves of $\mathcal{T}$ to t. See Figure 1 for an illustration. For the weighted max-flow/min-cut problem on a tree, $c = \infty$.

Definition 1 (s-t Weighted Max-Cut): Let $\mathcal{G}$ be a graph with source s, sink t, and non-negative edge weights. Let S and T be partitions of the nodes of $\mathcal{G}$ such that $s \in S, t \in T$. The s-t max-cut on $\mathcal{G}$ is the choice of S and T such that the sum of the weights of the edges from S to T is maximal.

The transformation in Figure 1 for the s-t max-cut problem on a tree is obtained by setting $c = -\infty$.

In general, max-flow and min/max-cut problems assume edge weights. However, in the stopping time setting, each node corresponds to a stopping option with a corresponding cost/reward, so we assume nodal weights. One problem setting can trivially be converted into the other without changing the optimal solution, as we show in Section IV.

III. Optimal Stopping Time Problem

Let T be the finite time-horizon and let $\mathbb{T} = \{t\}_{t=0}^{T}$ be the set of time-indices. Let $\Omega = \{\omega_i\}_{i=0}^{k-1}$ be a finite sample space and let $\mathcal{F} = (\mathcal{F}_t)_{t=0}^{T}$ be a filtration on Ω. Suppose without loss of generality that $\mathcal{F}_0 = \{\emptyset, \Omega\}$ and $\mathcal{F}_T = 2^{\Omega}$. Let $\mathbb{P}$ be a probability measure on Ω, defined at least on $\mathcal{F}_T$.

Definition 2 (Stopping Time): For a filtration $\mathcal{F}$ on Ω, a stopping time τ is a $\mathbb{T}$-valued random variable such that $\{\omega \in \Omega | \tau(\omega) = t\} \in \mathcal{F}_t, \forall t \in \mathbb{T}$. We say that τ is "adapted to" $\mathcal{F}$. Note that there is an abuse of terminology here, as τ is not a random process, but a random variable.

Definition 3 (Optimal Stopping Time): Let $\mathcal{S}$ be the set of all stopping times adapted to $\mathcal{F}$. Let $c = (c_t)_{t=0}^{T}$ be a random cost process adapted to $\mathcal{F}$. Let c_τ be the random variable such that $c_\tau(\omega) = c_{\tau(\omega)}(\omega), \forall \omega \in \Omega, \forall \tau \in \mathcal{S}$. An optimal stopping time $\tau^* \in \mathcal{S}$ is such that

$$E_{\mathbb{P}}[c_{\tau^*}] = \min_{\tau \in \mathcal{S}} E_{\mathbb{P}}[c_\tau] \, . \tag{1}$$

A. Representing Filtrations as Trees

In this section, we represent filtrations as trees, a special case of representing partially ordered sets (posets) as trees. See, for example, [8] for a set representation of trees, with applications to Game Theory.

Definition 4: For a σ-algebra $\mathcal{A}$ on Ω, a measurable set $A \in \mathcal{A}$ is an **atomic set** if $A \neq \emptyset$ and has no proper subset which is non-empty and measurable. Intuitively, atomic sets are the smallest non-empty measurable sets in a σ-algebra.

The sequence of σ-algebras are successive refinements, i.e., $\mathcal{F}_0 \subseteq \cdots \subseteq \mathcal{F}_T$, so we can represent $\mathcal{F}$ as a tree of depth T whose elements are its atomic sets. Any σ-algebra partitions Ω into equivalence classes.

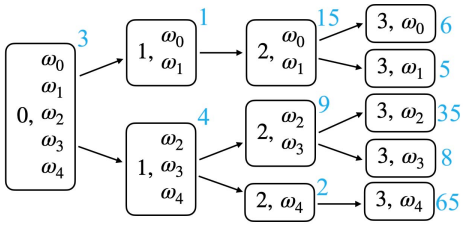

Fig. 2. Let $\Omega = \{\omega_i\}_{i=0}^{4}$. Consider filtration $\mathcal{F} = (\mathcal{F}_t)_{t=0}^{T=3}$, where $\mathcal{F}_0 = \{\emptyset, \Omega\}$, $\mathcal{F}_1 = \sigma(\{\omega_0, \omega_1\}, \{\omega_2, \omega_3, \omega_4\})$, $\mathcal{F}_2 = \sigma(\{\omega_0, \omega_1\}, \{\omega_2, \omega_3\}, \{\omega_4\})$, and $\mathcal{F}_{T=3} = 2^{\Omega}$. The graph shown is the tree representation of $\mathcal{F}$. Note how each $\omega \in \Omega$ corresponds to a leaf. Additionally, the labels next to each node represents the adapted process c: $c_0(\omega) = 3, \forall \omega \in \Omega$, $c_1(\omega_0) = c_1(\omega_1) = 1$, etc.

Definition 5: The **tree representation of** $\mathcal{F}$, denoted (V, E), is the tree with vertex set V and edge set E, where

$$V = \{(t, A) : t \in \mathbb{T}, A \text{ is an atomic set of } \mathcal{F}_t\}$$
$$E = \{((t-1, B), (t, A)) | (t, A) \in V, (t-1, B) \in V, A \subseteq B\} \, .$$

The root node is $(0, \Omega)$, and for all other nodes (t, A), its **parent** is the unique node $(t-1, B)$ such that $A \subseteq B$.

Since $\mathcal{F}^T = 2^{\Omega}$, Each $\omega \in \Omega$ corresponds to a leaf node in V. Define the **maximal path** corresponding to ω to be the unique path from the root $(0, \Omega)$ to the leaf $(T, \{\omega\})$.

Finally, for any random process X adapted to $\mathcal{F}$, we can assign the weight $X_t(\omega)$ to the node (t, A), where $\omega \in A, \forall t \in \mathbb{T}$. Since X is adapted, it does not matter which $\omega \in A$ is chosen; hence we will use $X_t(A)$ to refer to $X_t(\omega), \forall \omega \in A, \forall t \in \mathbb{T}$. In particular, this applies to the cost process c, an example of which is shown in Figure 2.

B. Tree Representation of the Optimal Stopping Problem

Proposition 1: Consider a discrete-time, finite-horizon optimal stopping time problem characterized by $(\Omega, \mathcal{F}, c, \mathbb{P})$.

Let (V, E) denote the tree representation of $\mathcal{F}$. For each $(t, A) \in V$, we associate the cost $L(t, A) = \mathbb{P}(A) c_t(A)$.

Then, the optimal stopping time problem is equivalent to finding a set $B \subseteq V$ that minimizes $\sum_{(t,A) \in B} L(t, A)$, subject to each maximal path having exactly one node in B.

Proof: A stopping time τ adapted to $\mathcal{F}$ can be represented as a subset of nodes $M \subseteq V$ at which to stop:

$$M = \{(t, A) \in V | \tau(\omega) = t, \forall \omega \in A\} \, .$$

Conversely, if $M \subseteq V$ contains exactly one node along each maximal path, then M defines a stopping time: each $\omega \in \Omega$ corresponds to a maximal path, and there is exactly one $(t, A) \in M$ s.t. $\omega \in A$. Thus, we can define the corresponding stopping time τ: if $(t, A) \in M$, $\tau(\omega) = t, \forall \omega \in A$. The tree structure forces τ to be adapted to $\mathcal{F}$.

Thus, any optimal stopping time τ^* will yield an optimal subset of nodes $M^* = \{(t, A) \in V | \tau^*(\omega) = t, \omega \in A\} \subseteq V$, and any optimal set $M^* \subseteq V$ will yield an optimal stopping time τ^*, where $\tau^*(\omega) = t$ s.t. $\omega \in A, (t, A) \in M^*$. ■

It is worth noting that the cost associated with node $(t, A) \in V$ is not just the realized value $c_t(A)$, but rather $c_t(A)$ weighted by $\mathbb{P}(A)$. The tree representation of an optimal stopping time problem is shown in Figure 3.

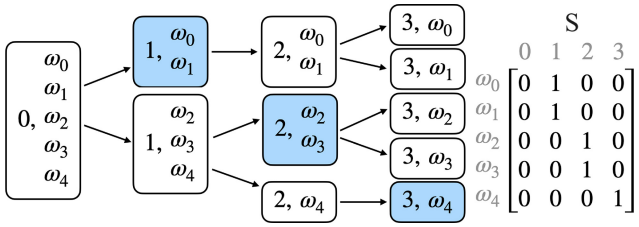

Fig. 3. Using the same filtration as in Figure 2, the stopping time τ corresponds to the colored nodes: $\tau(\omega_0) = \tau(\omega_1) = 1$, $\tau(\omega_2) = \tau(\omega_3) = 2$, and $\tau(\omega_4) = 3$. On the right is the binary matrix representation of τ. In each column, the values are all the same within each atomic set, capturing the measurability constraint. Across each row, there is exactly one entry which is 1, ensuring that exactly one stopping time $\tau(\omega)$ is assigned to each ω.

C. The Secretary Problem: an Example

An example of the optimal stopping time problem is the Secretary Problem (see, for example, [9]): a manager can only hire one out of n secretaries. The manager interviews the secretaries in a random order, and must make the decision to reject or hire each secretary right after the interview. Decisions are final, and if the manager hires a secretary, the remaining candidates will no longer be considered. The objective is to maximize the probability of choosing the best secretary.

After each interview, the manager gains more information about the relative rankings of the secretaries (assuming no ties). The time horizon is $T = n - 1$. Each time t corresponds to depth t in the filtration tree. Let the "reward" of node v be the probability of hiring the best secretary if stopping at v, weighted by $P(\{\omega \in v\})$. Let $a < b$ mean that secretary a ranks lower than secretary b. At $t = 0$ is the root, or the singleton ranking of secretary 1, with reward $\frac{1}{n} \cdot 1$ (this node represents Ω, and secretary 1 is ranked first in $(n-1)!$ rankings). At $t = 1$, there are two nodes, or scenarios: $1 < 2$ with reward $\frac{2}{n} \cdot \frac{1}{2} = \frac{1}{n}$ and $2 < 1$ with reward 0, and so on. Ω is the set of all rankings of the n secretaries, so the number of leaves is $|\Omega| = n!$. The size of the filtration tree is $\sum_{i=1}^{n} i!$.

IV. Optimal Stopping Time as a Min-Cut Problem

In this section, we show that the optimal stopping problem is equivalent to an s-t min-cut problem and frame it as a mixed-integer linear program (MILP), whose structure reveals some properties of s-t cut problems, which we will explore in Section V. Chen and Goldberg [4] showed the equivalence of the optimal stopping problem to the max-flow problem on a tree; here, we directly show equivalence to the min-cut problem on a tree, which is the dual of the max-flow problem.

Proposition 2 (Optimal Stopping is Equivalent to Min-Cut): Consider the node selection problem in Proposition 1. We add a source node s_0 and sink node s_1 to (V, E) as follows:

$$\tilde{V} = V \cup \{s_0, s_1\}$$

$$\tilde{E} = E \cup (s_0, (0, \Omega)) \cup \left(\bigcup_{\omega \in \Omega} ((T, \{\omega\}), s_1) \right)$$

$$\tilde{L}(e) = \begin{cases} \infty & v = s_1 \\ L(v) & \text{otherwise} \end{cases}, \quad \forall e = (u, v) \in \tilde{E} .$$

The node selection problem is equivalent to the min-cut problem on a tree $(\tilde{V}, \tilde{E})$ with capacities $\tilde{L}$.

Proof: Note that maximal paths of (V, E) correspond to s_0-s_1 paths in $(\tilde{V}, \tilde{E})$, and a min-cut will choose exactly one edge on each maximal path, giving a bijection between feasible sets with the same costs. Furthermore, no edge ending in s_1 will be chosen, and the desired result follows. ∎

The node selection problem in Proposition 1 naturally lends itself to a MILP with $|V|$ binary variables, each of which represents whether a node is included. However, we will formulate this problem with some redundant variables; this has a nicer interpretation from a probability space perspective.

Since Ω is finite, we can represent random variables on Ω as elements of $\mathbb{R}^{|\Omega|} = \mathbb{R}^k$, and denote the inner product $\langle X, Y \rangle = E_{\mathbb{P}}[XY]$ for random variables X and Y on the same probability space. In particular, $E_{\mathbb{P}}[X] = \langle \mathbf{1}_k, X \rangle$.

Furthermore, each σ-algebra $\mathcal{A}$ can be equivalently viewed as the subspace of all $\mathcal{A}$-measurable random variables, so that the conditional expectation $E[\,\cdot\,|\mathcal{A}]$ is simply the projection onto this subspace with respect to the aforementioned inner product. For our filtration $\mathcal{F}$, we will represent these projections as the linear operators P_t, where $P_t X = E[X|\mathcal{F}_t]$.

Proposition 3 (MILP Reformulation of the Optimal Stopping Problem): The stopping time τ^* is a solution to the optimal stopping problem in (1) if and only if the corresponding binary matrix S^* is an optimizer of the following MILP:

$$\begin{aligned} \min_{S \in \{0,1\}^{k \times (T+1)}} \quad & \sum_{t \in \mathbb{T}} \langle S_t, c_t \rangle \\ \text{s.t.} \quad & \sum_{t \in \mathbb{T}} S_t = 1_k \\ & (I - P_t) S_t = 0 \text{ for all } t \in \mathbb{T} . \end{aligned} \tag{2}$$

Proof: A stopping time τ can be represented as binary matrix $S \in \{0, 1\}^{k \times (T+1)}$, with the interpretation that $\tau(\omega) = t$ if and only if $S_t(\omega) = 1$. Here, we let $S_t(\omega)$ denote the entry in the ω^{th} row and t^{th} column. For a binary matrix S to represent a valid stopping time, we have two constraints:

1) Each row of S has exactly one non-zero entry; i.e., for each $\omega \in \Omega$, exactly one time index in $\mathbb{T}$ is chosen.
2) $P_t S_t = S_t, \forall t \in \mathbb{T}$, representing that S is adapted to $\mathcal{F}$; this is equivalent to $S_t = E[S_t|\mathcal{F}_t], \forall t \in \mathbb{T}$.

If $S \in \{0, 1\}^{k \times (T+1)}$ represents a stopping time τ, then

$$E_{\mathbb{P}}[c_\tau] = \sum_{t \in \mathbb{T}} \langle S_t, c_t \rangle .$$

These constraints and objective are incorporated into (2). ∎

Figure 3 shows an example of the matrix S.

Additionally, the linear program (LP) relaxation of this problem is exact, for the same reasons that min-cut programs can be solved as LPs. In the LP relaxation of (2), the constraint on S to be a binary matrix is replaced by $S \in [0, 1]^{k \times (T+1)}$. We will not give the full proof of the equivalence here, but at a high-level, unimodularity of the constraint matrices is sufficient for exactness of the LP relaxation of an MILP, and the constraints arising from a min-cut problem satisfy the unimodularity property. See, for example, [7] for details.

V. Equivalent Transformations for Min-Cut

We show through the particular structure of the MILP formulation (2) that an optimal stopping time problem with either a minimization or maximization objective and objective costs with arbitrary signs can be formulated into a min-cut problem on the same underlying tree structure.

A max-cut problem (corresponding to a maximization objective with non-negative rewards) on a tree can be transformed into an equivalent min-cut problem (corresponding to a minimization objective with non-negative costs) in $O(|V|)$ time, where $|V|$ is the number of nodes in the tree. In fact, we show that the min/max-cut problem on any tree can be formulated as an optimal stopping time problem, and so any s-t max-cut problem on a tree can be solved in linear time. In Section VI, we provide a $O(|V|)$ min-cut algorithm.

A. Transformation of Tree Capacities

We now find an equivalent formulation to (2). Define

$$D = \begin{bmatrix} P(\omega_0) & 0 & \dots & 0 \\ 0 & P(\omega_1) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & P(\omega_{k-1}) \end{bmatrix} \tag{3}$$

$$C = \begin{bmatrix} | & | & & | \\ c_0 & c_1 & \dots & c_T \\ | & | & & | \end{bmatrix}. \tag{4}$$

Proposition 4: Let $\Delta \in \mathbb{R}^k$ be a vector of increments s.t. $C+\Delta \mathbf{1}_{T+1}^T \geq 0$. In the underlying tree of (2), $P(\omega_i)\Delta_i$ is added to the capacity of the leaf u corresponding to $\omega_i \in \Omega$, as well as to those of u's ancestors. Replacing the objective of (2) with $\langle D(C + \Delta \mathbf{1}_{T+1}^T), S\rangle_F$ results in an equivalent problem, where $\langle \cdot, \cdot \rangle_F$ is the Frobenius inner product.

Proof:

$$\begin{aligned} \langle D\big(C + \Delta \mathbf{1}_{T+1}^T\big), S\rangle_F &= \langle DC, S\rangle_F + \langle D\Delta \mathbf{1}_{T+1}^T, S\rangle_F \\ &= \sum_{t\in\mathbb{T}} \langle S_t, c_t\rangle + \sum_{i=0}^{k-1} P(\omega_i)\Delta_i \sum_{t\in\mathbb{T}} S_t(\omega_i) \\ &= \sum_{t\in\mathbb{T}} \langle S_t, c_t\rangle + \sum_{i=0}^{k-1} P(\omega_i)\Delta_i, \end{aligned}$$

where the last equality is due to the constraint $S\mathbf{1}_k = \mathbf{1}_k$ in (2). Hence, the objective function of (2) is shifted by a constant, so the solution to (2) is unchanged. ∎

Figure 4 shows an example of an Optimal Stopping Time tree, with original capacities displayed inside the nodes.

Let us see how to transform the capacities of this tree:

$$\begin{aligned} D\big(C + \Delta \mathbf{1}_{T+1}^T\big) &= DC + D\Delta \mathbf{1}_{T+1}^T \\ &= \left[\begin{array}{c|c|c} P(\omega_0)c_0 & P(\omega_0)c_{1,0} & P(\omega_0)c_{2,0} \\ P(\omega_1)c_0 & P(\omega_1)c_{1,0} & P(\omega_1)c_{2,1} \\ P(\omega_2)c_0 & P(\omega_2)c_{1,1} & P(\omega_2)c_{2,2} \end{array}\right] \\ &\quad + \left[\begin{array}{c|c|c} P(\omega_0)\Delta_0 & P(\omega_0)\Delta_0 & P(\omega_0)\Delta_0 \\ P(\omega_1)\Delta_1 & P(\omega_1)\Delta_1 & P(\omega_1)\Delta_1 \\ P(\omega_2)\Delta_2 & P(\omega_2)\Delta_2 & P(\omega_2)\Delta_2 \end{array}\right]. \end{aligned} \tag{5}$$

The horizontal and vertical lines in the capacity matrix form regions, each of which corresponds to a node in Figure 4.

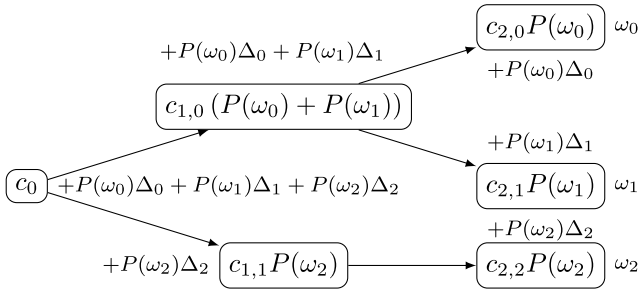

Fig. 4. An Optimal Stopping Time tree with the original capacities in each node. The leaves correspond to elements of Ω. The label next to each node corresponds to a modification of the node's capacity which altogether result in an equivalent min-cut problem as long as the $\Delta_{(\cdot)}$ are chosen so that the transformed capacities are all non-negative.

The capacity of each node is the sum of the capacities in its corresponding region in the matrix. The capacities inside the nodes correspond to entries of DC and the labels next to the nodes correspond to the entries of $D\Delta \mathbf{1}_{T+1}^T$ in (5).

We can generalize from the above example. First, we define some methods:

Definition 6: Let $\mathcal{T}, \mathcal{T}'$ be two trees over the same vertex and edge set, with root s. The nodal capacities may differ.

subtree(v): the tree rooted at node v. subtree$(v) \subseteq \mathcal{T}$.

leaves(v): the set of leaves in subtree(v). For a leaf u, leaves$(u) = \{u\}$. leaves(s) is simply the set of leaves of $\mathcal{T}$.

cap(v): the capacity of node v in $\mathcal{T}$.

cap$'(v)$: the capacity of node v in $\mathcal{T}'$.

For our setting, $\mathcal{T}$ is the underlying tree of a minimization optimal stopping time problem (a maximization problem would flip the signs of the capacities). For any $v \in \mathcal{T}$, cap(v) is possibly negative, and includes the probabilities of the nodes in leaves(v), as given by DC from (3) and (4).

Procedure 1: Let $\mathcal{T}$ be a tree rooted at s with vertex set V. Let $\Delta_v \in \mathbb{R}$ be such that cap$'(v) =$ cap$(v) + \Delta_v$. The following sequence of steps transforms a min/max-cut problem on $\mathcal{T}$ into an equivalent min-cut problem on $\mathcal{T}'$:

1) $\forall u \in$ leaves(s), let cap$'(u) =$ cap$(u) + \Delta_u$.
2) Iteratively process all children of a node before processing the node: for $v \in V$, let p be v's parent, and let Δ_p be initialized to 0. Update $\Delta_p = \Delta_p + \Delta_v$.

After processing each $v \in V$, cap$'(v) =$ cap$(v) + \sum_{u\in\text{leaves}(v)} \Delta_u$. For a valid min-cut formulation on $\mathcal{T}'$, $\Delta_u, \forall u \in$ leaves(s) must be such that cap$'(v) \geq 0, \forall v \in V$. Procedure 1 is a $O(|V|)$ DP algorithm, as each node is processed exactly once.

B. Generalization to s-t Min/Max-Cut on Any Rooted Tree

We will use Definition 6 in addition to the following:

Definition 7: Let $\mathcal{T}$ be a tree rooted at s. Let node $v \in \mathcal{T}$.

path(v): the path from s to v. Let $|\text{path}(v)|$ be the number of nodes in path(v). In particular, $|\text{path}(s)| = 1$.

Proposition 5: Let $\mathcal{T}$ be a tree rooted at s. There is an $O(|V|)$ algorithm which can transform an s-t max-cut problem on $\mathcal{T}$ to an equivalent s-t min-cut problem.

Proof: Let $S^u \in \{0,1\}^{|\text{path}(u)|}, \forall u \in$ leaves(s) be the vector indicating where on path(u) the min-cut is. The entries in

S^u which correspond to the same node must be equal. For instance, S^u_0, the first element of S^u, corresponds to $s, \forall u \in$ leaves(s), and so S^u_0 must agree, $\forall u \in$ leaves(s).

Let $c^u \in \mathbb{R}^{|\text{path}(u)|}, \forall u \in$ leaves(s) be the effective capacity of the nodes along path(u). Let c^u_i be the i^{th} entry of c^u; then, letting v be i^{th} node on path(v), $c^u_i = \frac{\text{cap}(v)}{|\text{leaves}(v)|}$.

The equivalent MILP formulation of the problem is

$$\begin{aligned} \min / \max_{S^u, \forall u \in \text{leaves}(s)} \quad & \sum_{u \in \text{leaves}(s)} \left(c^u\right)^T S^u \\ \text{s.t.} \quad & S^u \in \{0,1\}^{|\text{path}(u)|}, \forall u \in \text{leaves}(s) \\ & \mathbf{1}^T_{|\text{path}(u)|} S^u = 1, \forall u \in \text{leaves}(s) \\ & \text{constraints on } S^u \text{ to enforce } \mathcal{T}\text{'s structure.} \end{aligned} \tag{6}$$

Let $\Delta^u \in \mathbb{R}$. We see that, similar to the proof of Proposition 4, replacing c^u by $c^u + \Delta^u \mathbf{1}_{|\text{path}(u)|}$ in the objective of (6) does not change the solution, so Procedure 1 is still valid. ■

The s-t max-cut problem on a tree can be converted to an equivalent s-t min-cut problem in $O(|V|)$ time and solved in $O(|V|)$ time using the s-t min-cut algorithm in Section VI.

VI. Algorithm for s-t Min-Cut on Trees

We show that a DP algorithm finds the s-t max-flow/min-cut on trees. Let us now introduce some terminology used in Dinitz's algorithm [10], which is defined for graphs with edge capacities. However, our subsequent procedure assumes the optimal stopping setting: trees with nodal capacities.

Definition 8: Consider a graph $\mathcal{G}(V, E)$ with source s and sink t, where V is the vertex set and E is the edge set of $\mathcal{G}$.

dist(v): the number of edges in the shortest path from s to $v \in V$. Note that the distance from s to v does not take into account the edge weights along the s-v path.

level graph: a graph composed of a subset of E comprised of only the edges which are "leading away" from s: an edge $(u, v) \in E$ is in the level graph if dist(v) = dist(u) + 1. In other words, v is farther away from s than u is. In general, a graph may be different from its level graph.

saturated edge: an edge is saturated if the flow through that edge is already equal to its maximal value, the edge's capacity; i.e., no additional flow can be sent through the edge.

blocking flow: an s-t flow on $\mathcal{G}$ such that each s-t path has at least one saturated edge. Intuitively, this means that no extra flow can be sent along each s-t path.

As noted in [4], a blocking flow of a tree is a max-flow. We now provide one way of justifying this claim:

Proposition 6: For a tree $\mathcal{T}$ rooted at s whose leaves are connected to a sink t (as described in Section II-A), an s-t blocking flow on $\mathcal{T}$ is the s-t max-flow of $\mathcal{T}$.

Proof: At a high level, Dinitz's algorithm performs the following steps for a general graph $\mathcal{G}$:

Initialize $\mathcal{L}$ to be the level graph of $\mathcal{G}$.

1) Find a blocking flow of $\mathcal{L}$. Reduce the capacity of each edge in $\mathcal{L}$ by the flow through that edge. Remove edges from $\mathcal{L}$ which have 0 capacity (the saturated edges).
2) Introduce any new edges into $\mathcal{L}$ from $\mathcal{G}$ that previously were not eligible to be in $\mathcal{L}$, but become eligible after the removal of the saturated edges in Step 1.
3) Repeat steps 1 and 2 until $\mathcal{L}$ contains no s-t path.

For a tree $\mathcal{T}$, Dinitz's algorithm terminates after Step 1. This is because $\mathcal{T}$'s level graph is exactly $\mathcal{T}$; after Step 1, each leaf in $\mathcal{T}$ is no longer reachable from s. Thus, Dinitz's algorithm terminates and the blocking flow is optimal for $\mathcal{T}$. ■

A blocking flow can be found by greedily sending flow along the paths from the root to the leaves in a tree until each path has at least one saturated edge. This can be implemented via a DP algorithm. First, we introduce some notation. We will refer to *subtree* and *cap* as defined in Definition 6.

Definition 9: Consider a tree $\mathcal{T}$ rooted at s.

max_cap(v): the maximum flow through subtree(v).

cap_children(v): the sum of max_cap of the children of v. For all leaves $u \in \mathcal{T}$, cap_children(u) = cap(u). For all other nodes $v \in \mathcal{T}$, cap_children(v) is initialized to be 0.

Procedure 2 (s-t Min-Cut on Tree): Let $\mathcal{T}$ be a tree rooted at s. Instead of introducing a sink t, we will directly refer to $\mathcal{T}$'s leaves. We also assume nodal, instead of edge, capacities.

1) Beginning with the leaves of $\mathcal{T}$, iteratively process all children of a node v before v:
   a) The maximum flow through v cannot exceed the sum of the maximum flows through its children: set max_cap(v) = min(cap(v), cap_children(v)).
   b) Let p be the parent of v. Update cap_children(p): cap_children(p) = cap_children(p) + max_cap(v).

   Terminate after processing s.
2) At this point, a node v is saturated if max_cap(v) = cap(v). The min-cut is comprised of the saturated nodes closest to s. In other words, if v is in the min-cut, then no descendant of v can also be in the min-cut.

Step 1 takes $O(|V|)$ time. Step 2 requires a $O(|V|)$ procedure to identify the elements of the min-cut, via a graph traversal algorithm such as Depth- or Breadth-First Search. Thus the overall runtime of Procedure 2 is $O(|V|)$.

A. Sensitivity Analysis

Assume the optimal stopping time problem has already been solved via Procedure 2, and there is a change in cost for a certain node in the tree. Then, the new optimal stopping time solution may be obtained with a minimal amount of work, instead of running Procedure 2 on the entire tree with the incremental change.

We use Definitions 6, 7, and 9, in addition to the following:

Definition 10: Let $\mathcal{T}$ be the original underlying tree structure, and $\mathcal{T}'$ be identical to $\mathcal{T}$ except that the capacity for one node v changes. Let $\mathcal{G}$ be one of $\mathcal{T}$ or $\mathcal{T}'$.

min_cut$(\mathcal{G})$: the nodes in the min-cut of graph $\mathcal{G}$.

max_flow$(\mathcal{G})$: the maximum flow through $\mathcal{G}$.

Furthermore, when we say v is "above" min_cut$(\mathcal{T})$, we mean that no node along path(v) is part of min_cut$(\mathcal{T})$; instead, a subset of nodes in subtree(v) are necessarily part of min_cut$(\mathcal{T})$. Likewise, when we say v is "below" min_cut$(\mathcal{T})$, $\exists u \neq v \in$ min_cut$(\mathcal{T})$ s.t. u is an ancestor of v and is on path(v). Note that max_cap(u) and cap_children(u) for $u \neq v \in$ subtree(v) remain unchanged in $\mathcal{T}'$; however, these values can change for nodes along path(v).

1) Case 1: cap(v) < cap′(v):

- $v \in \text{min\_cut}(\mathcal{T})$: $\text{min\_cut}(\mathcal{T}')$ may differ from $\text{min\_cut}(\mathcal{T})$. $\text{max\_cap}(u)$ and $\text{cap\_children}(u)$ must be recalculated for each node u along $\text{path}(v)$.
- v is above $\text{min\_cut}(\mathcal{T})$: $\text{min\_cut}(\mathcal{T}) = \text{min\_cut}(\mathcal{T}')$. The maximal flow through $\text{subtree}(v)$ is restricted by the saturated nodes below v, so $\text{max\_flow}(\mathcal{T}) = \text{max\_flow}(\mathcal{T}')$.
- v is below $\text{min\_cut}(\mathcal{T})$: $\text{min\_cut}(\mathcal{T}) = \text{min\_cut}(\mathcal{T}')$. The ancestor u of v is still restricted by the same capacity in $\mathcal{T}'$, and so again $\text{max\_flow}(\mathcal{T}) = \text{max\_flow}(\mathcal{T}')$.

2) Case 2: cap(v) > cap′(v): Any ancestors of v which are unsaturated in $\mathcal{T}$ remain unsaturated in $\mathcal{T}'$ since the flow through them can only decrease or remain unchanged in $\mathcal{T}'$.

- $v \in \text{min\_cut}(\mathcal{T})$: v remains saturated in $\mathcal{T}'$ so $\text{min\_cut}(\mathcal{T}) = \text{min\_cut}(\mathcal{T}')$.
- v is above $\text{min\_cut}(\mathcal{T})$: Let $M = \text{subtree}(v) \cap \text{min\_cut}(\mathcal{T})$. If v becomes saturated in $\mathcal{T}'$, $\text{min\_cut}(\mathcal{T}') = (\text{min\_cut}(\mathcal{T}) \cup \{v\}) \setminus M$. Otherwise, for any $u \in \text{subtree}(v)$ which is above M, the flow through u can only decrease or stay the same and so u remains unsaturated. In this case, $\text{min\_cut}(\mathcal{T}) = \text{min\_cut}(\mathcal{T}')$.
- v is below $\text{min\_cut}(\mathcal{T})$: $\text{min\_cut}(\mathcal{T}')$ may differ from $\text{min\_cut}(\mathcal{T})$. $\text{max\_cap}(u)$ and $\text{cap\_children}(u)$ must be recalculated for each node u along $\text{path}(v)$.

Note that checking whether v is above or below the min-cut takes $O(T)$, where T is the time horizon, or the depth of $\mathcal{T}$. This can be accomplished by traversing $\text{path}(v)$ and checking if there are any ancestors of v which are saturated.

Recalculating $\text{max\_cap}(u)$ and $\text{cap\_children}(u)$ for any node $u \in \text{path}(v)$ also takes $O(T)$, since only the difference between $\text{max\_cap}(v)$ in $\mathcal{T}'$ and $\mathcal{T}$ is needed to update the two values. However, if $\text{min\_cut}(\mathcal{T}') \neq \text{min\_cut}(\mathcal{T})$, finding the saturated nodes closest to s in $\mathcal{T}'$ again takes $O(|V|)$.

We have thus shown that for an incremental change in capacity of node v, in the best case the solution remains unchanged and only an $O(T)$ traversal is required to find v's position relative to the min-cut. In the worst case, an $O(|V|)$ procedure is required to update the solution.

VII. Numerical Simulations

We implemented the procedures in this letter and solved the Secretary Problem, the details of which are explained in Section III-C. We compared the runtimes of Procedure 2 and the LP relaxation of the MILP (2) described in Section IV. Figure 5 displays the runtimes of solving the Secretary Problem for various problem sizes n, i.e., the number of secretaries. The size of the underlying filtration tree is also plotted for each n. The $O(|V|)$ DP algorithm is reasonably fast for all n we attempted, however, solving the LP becomes unmanageable as the filtration tree explodes in size.

We used Gurobi Optimizer to solve the LP. All simulations were performed on an Intel® Xeon® Platinum 8272CL CPU @ 2.60GHz processor.

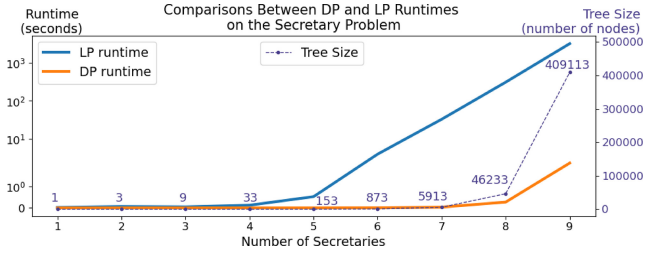

Fig. 5. Comparison of runtimes of the Linear Program and Dynamic Programming algorithm on the Secretary Problem, for various problem sizes. The scale for the tree sizes is shown on the right axis. The labels next to the data points are the corresponding tree sizes.

VIII. Conclusion

We have shown that any optimal stopping problem on a tree structure can be transformed into an s-t min-cut problem and solved in time linear in the underlying tree size. However, the tree size is often exponential in the time-horizon, in which case our algorithm may not be a practical solution. Future research directions may include adapting our results to develop sub-optimal solutions to large problems for which the exact solution is difficult to compute.

Furthermore, we have only considered the finite probability space setting. A potential further development would be to extend our results to the infinite-dimensional case.

Finally, our work may prove useful (especially the sensitivity analysis) in applications in which the optimal stopping time problem is the lower-level of a bilevel problem. For example, a seller decides on a pricing strategy subject to a consumer's decision of the optimal time to make a purchase.

References

[1] L. C. G. Rogers, "Monte carlo valuation of American options," *Math. Finance*, vol. 12, no. 3, pp. 271–286, 2002.
[2] M. B. Haugh and L. Kogan, "Duality theory and approximate dynamic programming for pricing American options and portfolio optimization," in *Handbooks in Operations Research and Management Science*, vol. 15. Amsterdam, The Netherlands: Elsevier, 2007, pp. 925–948.
[3] J. Tsitsiklis and B. Van Roy, "Optimal stopping of Markov processes: Hilbert space theory, approximation algorithms, and an application to pricing high-dimensional financial derivatives," *IEEE Trans. Autom. Control*, vol. 44, no. 10, pp. 1840–1851, Oct. 1999.
[4] D. A. Goldberg and Y. Chen, "Beating the curse of dimensionality in options pricing and optimal stopping," 2018, *arXiv:1807.02227*.
[5] Y. Y. Steven J. Benson, and X. Zhang, "Mixed linear and semidefinite programming for combinatorial and quadratic optimization," *Optim. Method. Softw.*, vol. 11, nos. 1-4, pp. 515–544, 1999.
[6] C. Dang, "Approximating a solution of the $s-t$ max-cut problem with a deterministic annealing algorithm," *Neural Netw.*, vol. 13, no. 7, pp. 801–810, 2000.
[7] V. V. Vazirani, *Approximation Algorithms*. Berlin, Germany: Springer, 2013. https://link.springer.com/content/pdf/bfm:978-3-662-04565-7/1?pdf=chapter%20toc
[8] C. Alós-Ferrer and K. Ritzberger, "Trees and decisions," *Econ. Theory*, vol. 25, no. 4, pp. 763–798, 2005.
[9] T. S. Ferguson, "Who solved the secretary problem?" *Statist. Sci.*, vol. 4, no. 3, pp. 282–289, 1989.
[10] Y. Dinitz, *Dinitz' Algorithm: The Original Version and Even's Version*. Berlin, Germany: Springer, 2006, pp. 218–240.