2-2017

Robust optimization for tree-structured stochastic network design

Xiaojian WU

Akshat KUMAR
Singapore Management University, akshatkumar@smu.edu.sg

Daniel SHELDON

Robust Optimization for Tree-Structured Stochastic Network Design

Xiaojian Wu[1] **Akshat Kumar**[2] **Daniel Sheldon**[3,4] **Shlomo Zilberstein**[3]

[1] Department of Computer Science, Cornell University, USA
[2] School of Information Systems, Singapore Management University, Singapore
[3] College of Information and Computer Sciences, University of Massachusetts Amherst, USA
[4] Department of Computer Science, Mount Holyoke College, USA
xw458@cornell.edu akshatkumar@smu.edu.sg {sheldon,shlomo}@cs.umass.edu

Abstract

Stochastic network design is a general framework for optimizing network connectivity. It has several applications in computational sustainability including spatial conservation planning, pre-disaster network preparation, and river network optimization. A common assumption in previous work has been made that network parameters (e.g., probability of species colonization) are precisely known, which is unrealistic in real-world settings. We therefore address the *robust river network design problem* where the goal is to optimize river connectivity for fish movement by removing barriers. We assume that fish passability probabilities are known only imprecisely, but are within some interval bounds. We then develop a planning approach that computes the policies with either high *robust ratio* or low *regret*. Empirically, our approach scales well to large river networks. We also provide insights into the solutions generated by our robust approach, which has significantly higher robust ratio than the baseline solution with mean parameter estimates.

1 Introduction

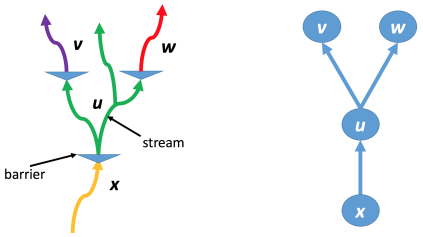
Many problems, such as influence maximization (Kempe, Kleinberg, and Tardos 2003), spatial and fish conservation planning (Sheldon et al. 2010; O'Hanley and Tomberlin 2005), and predisaster preparation (Schichl and Sellmann 2015) can be formulated as a variant of the stochastic network design problem. A stochastic network design problem (SNDP) is defined by a directed graph where each edge is either present or absent with some probability. Management actions can be taken to change the probabilities of edge presence. The goal is to determine which actions to take, subject to a budget, to optimize some outcome of the stochastic network over a time period. Several approaches to solve SNDPs have been shown to scale up to large networks (Chen, Wang, and Wang 2010; Kumar, Wu, and Zilberstein 2012; Wu, Sheldon, and Zilberstein 2014b; 2016).

An important assumption made in SNDPs is that the network parameters (e.g., probabilities of edge presence) are estimated accurately, which is not feasible in real world ecological domains due to noisy observations, model drift, climate change, and the diversity of species. To handle parameter uncertainty, researchers have formulated *robust* network design problems that include uncertain network probabilities (He and Kempe 2014; Chen et al. 2016). Recently, Kumar et al. (2016) also studied a robust conservation planning problem where the movement probabilities of species and sizes of habitats are not accurately specified. The robust network design problem we address differs from previous work, which does not allow management actions to modify interval parameters (e.g., edge probabilities). They only modify network structure, for example, by adding sources or nodes. In contrast, we allow management actions that can modify both interval bounds and network structure. As a result of the richer action space, it is unclear whether the sample average approximation (SAA) approach used in previous settings (Kumar et al. 2016) is applicable to our problem. To address these challenges, we develop a dynamic programming and mixed-integer programming based approach that can optimize connectivity without using SAA.

We study robust SNDPs for tree-structured river networks. The motivating application is the barrier removal problem (Neeson et al. 2015), where the goal is to decide which instream barriers to remove or repair to help fish move upstream and get access to their historical habitats. In this domain, the passage probability of a barrier can only be inaccurately estimated, and the new passage probability of a repaired barrier is even harder to estimate. Hence, we model the uncertainty in passage probability using well known interval bounds (Boutilier et al. 2003). We then develop a scalable algorithm to find the *robust policy* for barrier removal.

The robustness of a policy can be quantified by two correlated metrics: *robust ratio* (He and Kempe 2014; Chen et al. 2016) and *regret* (Boutilier et al. 2003; Kumar et al. 2016). Intuitively, assume that *given* a policy, nature chooses an *adversarial policy* that selects parameters within their interval bounds so as to either minimize the ratio between the values of the given policy and the adversarial policy (called *robust ratio*) or maximize the value difference between them (called *regret*). We develop a scalable algorithm to find a robust policy that maximizes the robust ratio by solving a bilevel optimization problem. We also show that, with minor modifications, our approach can be used to minimize regret.

The algorithm is based on a constraint generation procedure (Boutilier et al. 2003) that interleaves between two optimization steps. The *decision optimization* step finds a decision policy that maximizes the robust ratio when nature can



(a) River segments (b) Directed rooted tree

Figure 1: Encoding a river network as a directed rooted tree. Each color represents a contiguous region.

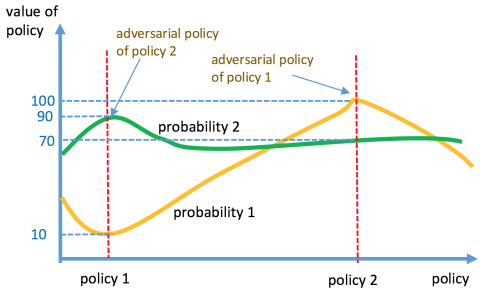


Figure 2: Illustration of robust ratio with X-axis showing different policies. For policy 1, the adversary chooses policy 2 and probability 1 (yellow curve) to minimize the robust ratio, which is 0.1. Similarly, the robust ratio of policy 2 is $\frac{7}{9}$, hence it is more robust than policy 1.

choose policies and probabilities from a given *limited* number of choices. In the second *ratio minimization* step, the best adversarial policy and probabilities are found for the selected decision policy and are added to the set of choices for nature. We provide a mixed integer linear programming formulation for the *decision optimization* problem. The *ratio minimization* problem is much harder; we develop an algorithm called rounded dynamic programming (RDP) by combining a dynamic programming algorithm and a rounding method and show that it is a fully polynomial time approximation schema (FPTAS). In experiments, we show that RDP performs nearly optimally as it selects the adversarial policy and probabilities. Our algorithm can find policies that are more robust than policies found by baseline methods with respect to both robustness metrics. We also provide insights on the robustness metrics by visualizing the solutions.

2 River Network Design

The problem is defined on a directed rooted tree $\mathcal{T}=(V,E)$ with a unique *root* denoted by s. Edges spread out from the root. A node v represents a contiguous region of the river network. It denotes a connected set of stream segments among which fish can move freely without passing any barriers. A node v is associated with a reward r_v which is proportional to the total amount of habitat in that region (e.g., the total length of all segments). An edge e encodes a river barrier. Fig. 1 shows how to encode a river network as a directed rooted tree. Each barrier is associated with a *passage probability*—the probability that a fish can pass the barrier. Before any repair action is taken, the probability is called the *initial passage probability* denoted by p_e. A finite set of *candidate actions* denoted by $A_e=\{0,1,...,m\}$ are available at e; an action i has cost $c_e(i)$, and, if taken, can raise passage probability to $p_{e|i}$. The action 0 is the *null action* with $p_{e|0}=p_e$ and zero cost. A policy π indicates which action is taken at each edge. The passage probability for a given policy is denoted by $p_{e|\pi}$. The accessibility of a node v denoted by $p_{s\rightsquigarrow v|\pi}$ is the probability that a fish passed all barriers on the path from s to v or $p_{s\rightsquigarrow v|\pi}=\prod_{e:\text{ on path from } s \text{ to } t} p_{e|\pi}$. A reward r_v can be collected only if a fish can reach v. The *value* of policy π, denoted by $z(\pi)$, is the total reward of nodes weighted by their accessibilities: $z(\pi)=\sum_{v\in V} p_{s\rightsquigarrow v|\pi} r_v$. We also call $z(\pi)$ the *objective value* to differentiate between other values assigned to π. The barrier removal problem (Wu, Sheldon, and Zilberstein 2014a) is to find a policy maximizing $z(\pi)$ subject to a budget constraint:

$$\arg\max_{\pi} z(\pi) \quad s.t. \quad c(\pi)\le\mathcal{B} \tag{1}$$

where $c(\pi)$ is the total cost of action taken for each edge in the network. Let $\mathcal{X}=\{\pi : c(\pi)\le\mathcal{B}\}$ denote the set of feasible policies.

Robust River Network Design The barrier removal problem is defined upon the assumption that all the passage probabilities are known. However, this is an unrealistic assumption. Often, in real world settings, it is not possible to accurately estimate such probabilities. Therefore, in our model only interval bounds are specified for different probabilities (Boutilier et al. 2003). Specifically, the passage probability for an edge e and action $i\in A_e$ can take any value within a given interval. That is, $p_{e|i}\in\mathcal{P}_{e|i}=[\underline{p}_{e|i},\overline{p}_{e|i}]$. Let $\boldsymbol{p}$ denote a vector of all probabilities $\boldsymbol{p}=(p_{e|i})_{e\in E,i\in A_e}$. Let the space of all the allowed probabilities $\boldsymbol{p}$ be denoted as $\mathcal{P}=\times_{e\in E,i\in A_e}\mathcal{P}_{e|i}$. Our goal is to find a policy π^{MRR} that *maximizes the robust ratio* as defined by Kouvelis and Yu (2013) and Chen et al. (2016):

$$\pi^{\text{MRR}}\in\operatorname*{argmax}_{\pi\in\mathcal{X}}\min_{\pi'\in\mathcal{X},\boldsymbol{p}\in\mathcal{P}}\frac{z(\pi;\boldsymbol{p})}{z(\pi';\boldsymbol{p})}. \tag{2}$$

In the outer maximization, the decision maker seeks a *decision policy* π that is robust relative to adversarial choices made by nature. In the inner minimization, nature adversarially chooses a policy π' and feasible parameters $\boldsymbol{p}$ (a *policy-parameter pair*) to minimize the ratio between the value of the decision policy π and the adversarial policy π' on this set of parameters. The optimal value of the adversary is called the *robust ratio* of policy π with respect to parameter space $\mathcal{P}$. A policy (such as π^{MRR}) that maximizes the robust ratio is called *MRR-optimal*, and the robust ratio of such a policy is called the *MRR-value*. Suppose π^{MRR} is MRR-optimal with MRR-value α: then π^{MRR} achieves at least α fraction of the optimal reward for any parameter setting $\boldsymbol{p}\in\mathcal{P}$. Fig. 2 illustrates the concept.

Algorithm 1 Robust Policy Optimization

1: Initialize $C = \{(\pi'_0, \boldsymbol{p}_0)\}$ and $T = 1$.
2: **Decision Optimization:** obtain π_T by solving:

$$U = \max_{\pi} \min_{(\pi', \boldsymbol{p}) \in C} z(\pi; \boldsymbol{p}) / z(\pi'; \boldsymbol{p}) \tag{3}$$

3: **Adversary Optimization:** obtain the adversarial policy-parameter pair $(\pi'_T, \boldsymbol{p}_T)$ with respect to π_T by solving:

$$L = \min_{(\pi', \boldsymbol{p}) \in C} z(\pi_T; \boldsymbol{p}) / z(\pi'; \boldsymbol{p}). \tag{4}$$

4: if $U - L \leq threshold$, return π_T. Otherwise set $C = C \cup \{(\pi'_T, \boldsymbol{p}_T)\}$, increment T, and go to step 2.

3 Our Method

We develop an iterative method (Algorithm 1) to solve Problem (2) using constraint generation (Boutilier et al. 2003). The high-level idea is to interleave two optimization problems. First, in the *decision optimization problem*, the decision maker finds the best decision policy π_T relative to a *limited* adversary, who can only pick policy-parameter pairs from the finite set C. Then, the adversary selects a new policy-parameter pair to minimize the robust ratio with respect to the current decision policy π_T. The decision player's value U is an upper bound on the MRR-value, because the adversary is limited to a finite subset of policy-parameter pairs. The adversary's optimal value L is a lower bound on the MRR-value. When $U = L$, we have an MRR-optimal decision policy. By allowing a small gap between the two bounds, we can find a nearly MRR-optimal policy. The set C is initialized with an arbitrary policy and probabilities.

3.1 The Decision Optimization Problem

The goal of Problem (3) is to find a decision policy that maximizes the robust ratio with respect to the limited adversary. Fig. 3 presents a mixed-integer linear program (MILP) to solve this problem building on techniques from (Neeson et al. 2015). The variable M encodes the MRR-value. The inner minimization is replaced by inequality constraints (6) on M. The continuous variable $z_{\boldsymbol{p}}$ encodes the objective value of the decision policy for probability setting $\boldsymbol{p}$ by (7). $z(\pi'; \boldsymbol{p})$ is a constant for each policy-parameter pair $(\pi'; \boldsymbol{p}) \in C$. x_e^i is a binary decision variable indicating whether action $i \in A_e$ is applied to e $(= 1)$ or not $(= 0)$. Constraint (8) enforces that one and only one action is taken at each edge, and (9) is the budget constraint.

The constraint set $\Omega(\boldsymbol{p}, x)$ defined in (12)–(18) forces $z_{\boldsymbol{p}}$ to be the objective value of π under probability setting $\boldsymbol{p}$. The variable $\alpha_v^{\boldsymbol{p}}$ encodes the accessibility of node v. The root node has accessibility 1 by (13). $\Pi(v)$ denotes the *parent* of node v. Recall that each node has at most one parent. The variable $\lambda_{v,i}^{\boldsymbol{p}}$ encodes the increment in the accessibility of node v if an action $i \in A_{\Pi(v),v}$ is applied to edge $(\Pi(v), v)$. In (14), the accessibility of v equals to the cumulative passability when no action is taken on edge $(\Pi(v), v)$ (the term $\alpha_{\Pi(v)}^{\boldsymbol{p}} p_{\Pi(v)v}$) plus the total increment (the term $\sum_{i \in A_{\Pi(v)v}} \lambda_{v,i}^{\boldsymbol{p}}$). Actually, at most one action can be taken, so only one $\lambda_{v,i}^{\boldsymbol{p}}$ will be nonzero in the summation. The increment $\lambda_{v,i}^{\boldsymbol{p}}$ is nonzero only if $x_{\Pi(v)v}^i$ is 1 by (15), and can be at most $(p_{\Pi(v)v|i} - p_{\Pi(v)v})\, \alpha_{\Pi(v)}^{\boldsymbol{p}}$ by (16), which is exactly the increment when action i is taken.

$$\max M \tag{5}$$

$$M \leq \frac{z_{\boldsymbol{p}}}{z(\pi'; \boldsymbol{p})} \qquad \forall (\pi'; \boldsymbol{p}) \in C \tag{6}$$

$$z_{\boldsymbol{p}} \in \Omega(\boldsymbol{p}, x) \qquad \forall (\pi'; \boldsymbol{p}) \in C \tag{7}$$

$$\sum_{i \in A_e} x_e^i = 1 \qquad \forall e \in E \tag{8}$$

$$\sum_{e \in E} \sum_{i \in A_e} c_i x_i \leq \mathcal{B} \tag{9}$$

$$x_e^i \in \{0, 1\} \quad \forall e \in E, \forall i \in A_e \tag{10}$$

Constraint set $\Omega(\boldsymbol{p}, x)$ (11)

$$z_{\boldsymbol{p}} = \sum_{v \in V} \alpha_v^{\boldsymbol{p}} r_v \tag{12}$$

$$\alpha_s^{\boldsymbol{p}} = 1 \tag{13}$$

$$\alpha_v^{\boldsymbol{p}} = \alpha_{\Pi(v)}^{\boldsymbol{p}} p_{\Pi(v)v} + \sum_{i \in A_{\Pi(v)v}} \lambda_{v,i}^{\boldsymbol{p}} \qquad \forall v \in V/\{s\} \tag{14}$$

$$\lambda_{v,i}^{\boldsymbol{p}} \leq x_{\Pi(v)v}^i \qquad \forall v \in V/\{s\}, \forall i \in A_{\Pi(v)v} \tag{15}$$

$$\lambda_{v,i}^{\boldsymbol{p}} \leq (p_{\Pi(v)v|i} - p_{\Pi(v)v})\, \alpha_{\Pi(v)}^{\boldsymbol{p}} \quad \forall v \in V/\{s\}, \forall i \in A_{\Pi(v)v} \tag{16}$$

$$\alpha_v^{\boldsymbol{p}} \in [0, 1] \qquad \forall (\pi', \boldsymbol{p}) \in C, \forall v \in V \tag{17}$$

$$\lambda_{e,i}^{\boldsymbol{p}} \in [0, 1] \qquad \forall (\pi', \boldsymbol{p}) \in C, \forall e \in E, \forall i \in A_e \tag{18}$$

Figure 3: Mixed integer linear program to maximize the robust ratio for a given set C

3.2 The Adversary Optimization Problem

In the adversary optimization step, we wish to solve Problem (4) to find a policy-parameter pair $(\pi'^*, \boldsymbol{p}^*)$ to minimize the robust ratio with respect to the current decision policy.

Here is our main result.

Theorem 1. *There is an FPTAS for problem (4). It finds a policy-parameter pair with robust ratio at most* $(1+\epsilon)OPT$ *in time* $O(\frac{n^4}{\mu^2})$ *where* $\mu = \frac{\epsilon}{2+\epsilon}$*,* n *is the number of nodes in the tree, and* OPT *is the optimal value of (4).*

The FPTAS only approximately minimizes the objective, so the value $\hat{L}$ it achieves not a lower bound in in Algorithm 1. However, the approximation guarantee implies that $L = \frac{\hat{L}}{1+\epsilon}$ is a lower bound.

In the rest of this section, we prove Theorem 1 (proofs of some auxiliary results are left in appendix). We first propose a dynamic programming (DP) algorithm for problem (4), but this takes exponential time. We then develop a rounding strategy to reduce the running time to polynomial time and prove that this is an FPTAS. This basic idea is originally used for the barrier removal problem (1) (Wu, Sheldon, and Zilberstein 2014a). The adversary optimization problem here is more complex as the adversary tries to simultaneously minimize the value of decision policies and maximizes the value of adversarial policies. To guarantee the approximation rate, we round these two values distinctly.

To simplify the presentation, we assume without loss of generality the following:

Assumption 1. *Each node $u \in \mathcal{T}$ has at most two children.*

Any problem instance can be converted to satisfy this assumption (Wu, Sheldon, and Zilberstein 2014a). Our first lemma restricts the space of parameters to be considered.

Lemma 1. *There exists an optimal policy-parameter pair $(\pi'^*, \boldsymbol{p}^*)$ for Problem (4) with the following property. Suppose π'^* takes action i and the decision policy π takes action j on edge e. If $j \neq i$, then $p^*_{e|i} = \overline{p}_{e|i}$ and $p^*_{e|j} = \underline{p}_{e|j}$. Otherwise, $p^*_{e|i}$ is either $\underline{p}_{e|i}$ or $\overline{p}_{e|i}$.*

Lemma 1 guarantees that the optimal adversary probability is either the upper or lower bound of the interval.

Policy-Parameter Actions and Optimization First, we redefine problem (4) in the following way so that it is amenable to dynamic programming.

Let π be fixed. The new optimization problem is the same as the river network design problem (1) except that its objective is the robust ratio $\frac{z(\pi;\boldsymbol{p})}{z(\pi';\boldsymbol{p})}$ and its actions encode both the actions and parameters of the adversary.

We define a finite set of *policy-parameter actions* A^p_e for each edge, which encode choices made by the adversary for edge e, including both the action taken and the probability setting for each available action. A policy-parameter action is a vector $(\boldsymbol{i}^a_e, \boldsymbol{p}_{e|0}, ..., \boldsymbol{p}_{e||A_e|})$ taking value in $A^p_e = A_e \times \prod_{j \in A_e} \{\underline{p}_{e|j}, \overline{p}_{e|j}\}$. $\boldsymbol{i}^a_e$ specifies the action that the adversary takes at e. $\boldsymbol{p}_{e|j}$ specifies the passage probability on e for action j. It is easy to see from Lemma 1 that a given policy-parameter action need only consider $\underline{p}_{e|j}$ and $\overline{p}_{e|j}$ as possible values for $\boldsymbol{p}_{e|j}$ without sacrificing optimality. In addition, Lemma 1 allows us to eliminate certain policy-parameter actions from consideration. For example, if $A_e = \{0, 1\}$ and the decision policy π takes action 1, A^s_e only needs to include 3 policy-parameter actions

$$(0, \overline{p}_{e|0}, \underline{p}_{e|1}), \;\; (1, \underline{p}_{e|0}, \underline{p}_{e|1}), \;\; (1, \underline{p}_{e|0}, \overline{p}_{e|1})$$

More generally, we have

Corollary 1. *For a fixed π, only $|A_e| + 1$ actions in A^s_e are needed to compute $(\pi'^*, \boldsymbol{p}^*)$.*

In summary, the choice of a policy-parameter action for each edge to minimize the robust ratio gives the optimal policy-parameter pair $(\pi'^*, \boldsymbol{p}^*)$ for problem (27).

Dynamic Programming We now present a dynamic programming algorithm to solve this new problem with policy-parameter actions.

In a rooted directed tree, each node u corresponds to a subtree $\mathcal{T}_u$. Define π_u (or π'_u) to be the subset of π (or π') that only includes actions for edges within $\mathcal{T}_u$, and define $\boldsymbol{p}_u$ to be the subset of $\boldsymbol{p}$ including probabilities only in $\mathcal{T}_u$. Define $z_u(\pi_u; \boldsymbol{p}_u)$ to be the objective value of policy π_u on subtree $\mathcal{T}_u$ with probability vector $\boldsymbol{p}_u$ pretending that u is the overall root, i.e., $z_u(\pi_u; \boldsymbol{p}_u) = \sum_{t \in \mathcal{T}_u} p_{u \rightsquigarrow t|\pi} r_t$. Similarly, $z_u(\pi'_u; \boldsymbol{p}_u)$ is the value of π'_u for $\mathcal{T}_u$. The following recurrences calculate both values for a given $(\pi_u; \boldsymbol{p}_u)$

$$z_u(\pi'_u; \boldsymbol{p}_u) = r_u + p_{uv|\pi'_u} z_v(\pi'_v; \boldsymbol{p}_v) + p_{uw|\pi'_u} z_w(\pi'_w; \boldsymbol{p}_w) \quad (19)$$

$$z_u(\pi_u; \boldsymbol{p}_u) = r_u + p_{uv|\pi_u} z_v(\pi_v; \boldsymbol{p}_v) + p_{uw|\pi_u} z_w(\pi_w; \boldsymbol{p}_w) \quad (20)$$

The DP table of subtree $\mathcal{T}_u$ is indexed by pairs (z^a_u, z^d_u), where z^a_u represents an objective value of an adversary policy and z^d_u represents an objective value of the (fixed) decision policy on that subtree. The table includes only pairs that are achievable by some probability vector $\boldsymbol{p}_u$ and adversary policy π'_u for subtree $\mathcal{T}_u$, that is, $z^d_u = z_u(\pi_u; \boldsymbol{p}_u)$ and $z^a_u = z_u(\pi'_u, \boldsymbol{p}_u)$. Let $\Phi(z^a_u, z^d_u) = \{(\pi'_u, \boldsymbol{p}_u) \mid z_u(\pi'_u; \boldsymbol{p}_u) = z^a_u, z_u(\pi_u; \boldsymbol{p}_u) = z^d_u\}$ be the set of all policy-parameter pairs that map to a pair of objective values (z^a_u, z^d_u). For the entry of the table indexed by (z^a_u, z^d_u), we record only the *minimum-cost* adversary policy, and the minimum cost (denoted by mc) it achieves:

$$\mathrm{mc}(z^a_u, z^d_u) = \min_{(\pi', \boldsymbol{p}) \in \Phi(z^a_u, z^d_u)} c(\pi'_u) \quad (21)$$

The DP tables for all subtrees can be calculated recursively from leaf nodes toward the root s in the following way. First, the table at a leaf node contains a single tuple with cost 0 because the subtree contains only the leaf node. Consider a node u with two children v and w. We can build the DP table at u if we have the DP tables of v and w by computing all achievable objective-value pairs at u and their minimum costs. From each pair (z^a_v, z^d_v) at v and each pair (z^a_w, z^d_w) at w, policy-parameter pairs $(\pi'_v, \boldsymbol{p}_v)$ and $(\pi'_w, \boldsymbol{p}_w)$ can be extracted. For each policy-parameter action $(i^a_{uv}, \boldsymbol{p}_{uv})$ on edge (u, v) and each policy-parameter action $(i^a_{uw}, \boldsymbol{p}_{uw})$ on edge (u, w), a new pair $(\pi'_u, \boldsymbol{p}_u)$ at u can be built, with which we can compute a pair (z^a_u, z^d_u) using recurrences (19) and (20). The cost of this new pair is

$$c(i^a_{uv}) + c(i^a_{uw}) + \mathrm{mc}(z^a_v, z^d_v) + \mathrm{mc}(z^a_w, z^d_w) \quad (22)$$

The same pair may be generated multiple times, but only the minimum cost is recorded.

Once all DP tables are computed, the optimal solution can be extracted from the table at s by finding a tuple

$$(z^{a*}_s, z^{d*}_s) \in \arg \min_{\mathrm{mc}(z^a_s, z^d_s) \leq \mathcal{B}} \frac{z^d_s}{z^a_s}$$

The pair $(\pi'^*, \boldsymbol{p}^*)$ associated with the tuple minimizes the objective.

Unfortunately, the table size grows exponentially with the height of the node in the tree. We next introduce a rounding strategy to make the algorithm scalable.

Rounding We define rounded value functions $\hat{z}_u(\pi'; \boldsymbol{p})$ and $\hat{z}_u(\pi; \boldsymbol{p})$ for subtree u and introduce the following recurrences for *rounded* value functions:

$$\hat{z}_u(\pi'_u; \boldsymbol{p}_u) = K_u \left\lfloor \frac{r_u + p_{uv|\pi'_u} \hat{z}_v(\pi'_v; \boldsymbol{p}_v) + p_{uw|\pi'_u} \hat{z}_w(\pi'_w; \boldsymbol{p}_w)}{K_u} \right\rfloor \quad (23)$$

$$\hat{z}_u(\pi_u; \boldsymbol{p}_u) = K_u \left\lceil \frac{r_u + p_{uv|\pi_u} \hat{z}_v(\pi_v; \boldsymbol{p}_v) + p_{uw|\pi_u} \hat{z}_w(\pi_w; \boldsymbol{p}_w)}{K_u} \right\rceil \quad (24)$$

where K_u is an user defined *rounding parameter*. Intuitively, values are rounded and grouped into discrete bins, which reduces the number of pairs in the DP table. The following theorem states that for any given policy-parameter pair, the rounded objective values are not too far from the true values.

Theorem 2. *Let $\mu > 0$. If we set $K_u = \mu r_u$, for any $(\pi'_u, \boldsymbol{p}_u)$ and any π_u, we have*

$$z_u(\pi'_u;\boldsymbol{p}_u) - \hat{z}_u(\pi'_u;\boldsymbol{p}_u) \le \sum_{t\in\mathcal{T}_u} p_{u\rightsquigarrow t|\pi'_u} K_t = \mu z_u(\pi'_u;\boldsymbol{p}_u) \quad (25)$$

$$\hat{z}_u(\pi_u;\boldsymbol{p}_u) - z_u(\pi_u;\boldsymbol{p}_u) \le \sum_{t\in\mathcal{T}_u} p_{u\rightsquigarrow t|\pi_u} K_t = \mu z_u(\pi_u;\boldsymbol{p}_u) \quad (26)$$

$$z_u(\pi'_u;\boldsymbol{p}_u) \ge \hat{z}_u(\pi'_u;\boldsymbol{p}_u) \quad (27)$$

$$\hat{z}_u(\pi_u;\boldsymbol{p}_u) \ge z_u(\pi_u;\boldsymbol{p}_u) \quad (28)$$

Proof sketch. Intuitively, in (23), the floor rounding operation at a node t reduces the value by at most K_t, which is discounted by probability $p_{u\rightsquigarrow t|\pi'}$. Therefore, we have (25) and (27). In (24), the ceiling rounding operation at a node t introduces an increment bounded by K_t, which is discounted by $p_{u\rightsquigarrow t|\pi'}$. Therefore, we have (26) and (28). □

The rounded dynamic programming (RDP) algorithm works the same as the DP algorithm except that instead of keeping a list of (z^a_u, z^d_u) in the table of u, a list of *rounded pairs* denoted by $(\hat{z}^a_u, \hat{z}^d_u)$ are kept, which are calculated by recurrences (23) and (24). Each rounded pair is associated with the minimum cost to achieve it and the correspondent policy-parameter pair. Intuitively, since multiple z^a_us (or z^d_us) are rounded into the same $\hat{z}^a_u$ (or $\hat{z}^d_u$), the size of the table is reduced. It can be shown that RDP can find

$$(\pi'^r, \boldsymbol{p}^r) \in \arg\min_{\pi',\boldsymbol{p}} \frac{\hat{z}(\pi;\boldsymbol{p})}{\hat{z}(\pi';\boldsymbol{p})} \quad (29)$$

We show that $(\pi'^r, \boldsymbol{p}^r)$ is a good approximation to the optimal policy-parameter pair $(\pi'^*, \boldsymbol{p}^*)$. That is, it is within $(1+\epsilon)$ optimal if μ is set properly. Specifically,

Theorem 3. *If $\mu = \frac{\epsilon}{2+\epsilon}$, we have*

$$OPT = \frac{z(\pi;\boldsymbol{p}^*)}{z(\pi'^*;\boldsymbol{p}^*)} \le \frac{z(\pi;\boldsymbol{p}^r)}{z(\pi'^r;\boldsymbol{p}^r)} \le (1+\epsilon)OPT$$

Proof. By (25) and (26), for any $(\pi', \boldsymbol{p})$, we have

$$\frac{\hat{z}(\pi;\boldsymbol{p})}{\hat{z}(\pi';\boldsymbol{p})} \le \frac{(1+\mu)z(\pi;\boldsymbol{p})}{(1-\mu)z(\pi';\boldsymbol{p})} = (1+\epsilon)\frac{z(\pi;\boldsymbol{p})}{z(\pi';\boldsymbol{p})}$$

Since $(\pi'^r, \boldsymbol{p}^r)$ produces the minimum ratio for rounded value functions (24) and (23), we have

$$\frac{\hat{z}(\pi;\boldsymbol{p}^r)}{\hat{z}(\pi'^r;\boldsymbol{p}^r)} \le \frac{\hat{z}(\pi;\boldsymbol{p}^*)}{\hat{z}(\pi'^*;\boldsymbol{p}^*)} \le (1+\epsilon)\frac{z(\pi;\boldsymbol{p}^*)}{z(\pi'^*;\boldsymbol{p}^*)}$$

By (27) and (28), we have

$$\frac{z(\pi;\boldsymbol{p}^r)}{z(\pi'^r;\boldsymbol{p}^r)} \le \frac{\hat{z}(\pi;\boldsymbol{p}^r)}{\hat{z}(\pi'^r;\boldsymbol{p}^r)}$$

Thus, the theorem is proved. □

Runtime Analysis In Theorem 3, we see that the K_u values of affect the approximation rate. Now, we analyze the dependence of the RDP algorithm running time on these values. First, we make the following assumption.

Assumption 2. *There are two constants m and M independent of $|V|$ such that $m \le r_u \le M$ for all $u \in V$.*

The assumption is reasonable because rewards represent habitat areas of stream segments, which do not increase or decrease as the number of segments increases.

Let the number of different values of $\hat{z}^a_u$ and $\hat{z}^d_u$ in the table at u be m^a_u and m^d_u. We have

Lemma 2. *If $K_u = \mu r_u$, we have*

$$m^a_u = O\Big(\frac{n_u}{\mu}\Big), \quad m^d_u = O\Big(\frac{n_u}{\mu}\Big)$$

where n_u is the number of nodes in subtree $\mathcal{T}_u$.

Proof. Since $\hat{z}(\pi'_u;\boldsymbol{p}_u)$ is upper-bounded by $z(\pi'_u;\boldsymbol{p}_u) \le n_u \cdot M$, the number of different rounded values with K_u is $m^a_u \le \frac{n_u \cdot M}{K_u} \le \frac{n_u \cdot M}{\mu m} = O(\frac{n_u}{\mu})$. Similarly, $\hat{z}(\pi;\boldsymbol{p})$ is upper-bounded by $(1+\mu)z(\pi_u;\boldsymbol{p}_u) \le (1+\mu)n_u M$, so $m^d_u = O(\frac{n_u}{\mu})$ as well. □

Define $T(n_u)$ to be the running time for subtree u, which is calculated by recurrence

$$T(n_u) = O(m^a_v m^d_v m^a_w m^d_w) + T(n_v) + T(n_w)$$

Together with Lemma 2, it can be shown that

Theorem 4. $T(n_u) = O(\frac{n_u^4}{\mu^2})$.

Thus, the running time of the RDP algorithm is $O(\frac{n^4}{\mu^2})$ where n is the number of nodes in the directed rooted tree. Combining Theorems 3 and 4, Theorem 1 is proved.

4 Other Criterion of Robustness

A slightly different way to quantify robustness is to use *regret* (Kumar et al. 2016; Boutilier et al. 2003). The policy that minimizes the regret is defined by

$$\pi^{\mathrm{MR}} \in \arg\min_{\pi:c(\pi)\le\mathcal{B}} \max_{\pi':c(\pi')\le\mathcal{B}} z(\pi';\boldsymbol{p}) - z(\pi;\boldsymbol{p}) \quad (30)$$

The robust ratio and the regret are correlated as

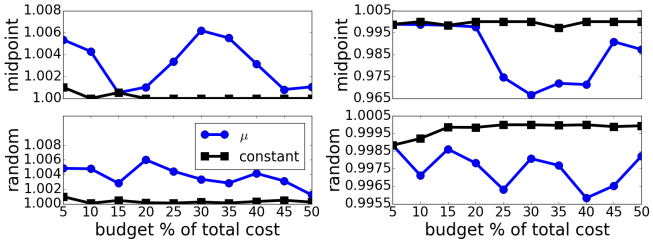
$$\frac{z(\pi;\boldsymbol{p})}{z(\pi';\boldsymbol{p})} = 1 - \frac{z(\pi';\boldsymbol{p}) - z(\pi;\boldsymbol{p})}{z(\pi';\boldsymbol{p})}$$

The robust ratio is in some way the scaled version of the regret. In experiments, we show that π^{MRR} also produces small regret compared to policies computed by other baseline methods. Our algorithm with minor modifications can find a nearly optimal π^{MR} empirically.

5 Experiments

We use data from the CAPS project (McGarigal et al. 2011) for the river networks in Massachusetts and synthetically define the missing parameters from the data. The data provides the point estimates of the initial passability probabilities. We use the method in (Kumar et al. 2016) to define the intervals of initial passage probabilities before taking actions. The interval of an initial passage probability is $[p - \beta p, p + \beta p]$ where p is an point estimate and β is a parameter controlling the interval sizes.

The data contains two types of barriers: culverts and dams. The point estimates for culverts provided by the data



(a) RDP for robust ratio (b) RDP for regret

Figure 4: Approximate qualities for different algorithm configurations with $\beta = 0.3$. X-axis: budget sizes. Y-axis: $\frac{value}{OPT}$ where OPT is the optimal value produced by the DP algorithm. Value of random policies is an average of 10 runs.

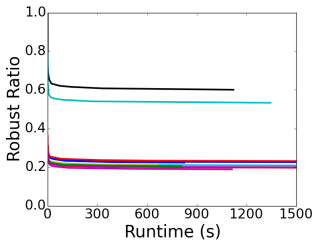
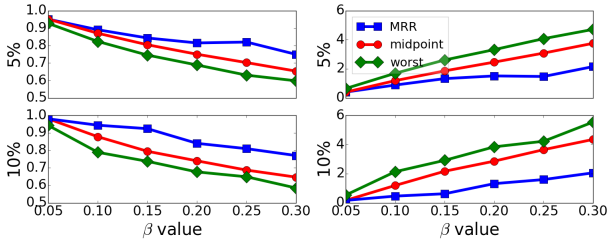


Figure 5: Robust ratio for different K values with β=0.3. From top to bottom, curves are for "midpoint" and "worst" policies, and 10 random policies.

are mostly in the range $[0.8, 0.9]$. A typical action that removes a culvert raises its passage probability to 1.0 and costs \$100,000. Most of the point estimates for dams are less than 0.2. A typical action to repair a dam costs \$173,030, and shifts its probability interval to $[p' - \beta p', p' + \beta p']$ where $p' = p +$ a random value in [0.5, 0.9] . The cost estimates are based on a study by Neeson et al. (2015). All intervals are truncated to fit within $[0, 1.0]$.

We compare our algorithm against two baseline methods: a "midpoint" policy is obtained by solving problem (1) and assuming true passage probabilities being the mid-point values of the intervals; a "worst" policy is obtained by solving problem (1) and conservatively assuming true passage probabilities being the lower bounds of the intervals. The policy calculated by our algorithm is the "MRR" policy.

Approximate Rate of the RDP Algorithm First, we evaluate the approximation rates of the RDP algorithm for problem (4), and of a *modified RDP algorithm* for solving the inner maximization problem of (30) on a small network of only 22 nodes. The DP algorithm runs out of memory on networks of larger sizes. The results are shown in Fig. 4. We set K_u in two different ways—$\epsilon = 0.1$ (denoted by "μ") and $K_u = 5$ (denoted by "constant"). Setting $K_u = 5$ makes the algorithm about 20 times faster than setting $\mu = 0.1$ and 100–600 times faster than DP. Note that robust ratios produced by our algorithm are greater than OPT and regrets are smaller than OPT. From the figures, we see that the (modified) RDP algorithm produces nearly optimal policy-parameter pairs. In the rest of experiments, we do not show



(a) Robust Ratio (b) Regret

Figure 6: Robust ratio and regret ($\times 10^5$) for three type of policies under different β and budget sizes of 5% and 10% .

the results of the modified algorithm to solve problem (30).

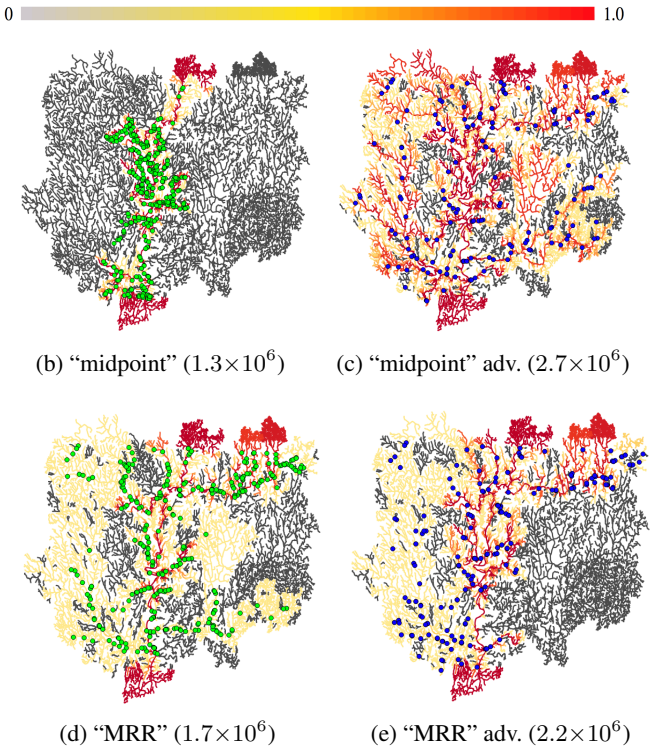
We test on a larger network of 2028 culverts and 166 dams to see what value of K, when we set K_u=K, is sufficiently large for the RDP algorithm to produce good robust ratios. The optimal objective value is not available on this network. The results are shown in Fig. 5. We see that robust ratios converge within 2 minutes for all testing policies, and random policies are much worst than two baseline policies. The value of K in the convergence area implies that it is sufficient to produce near-optimal solutions.

Robustness Comparison On the same network, we compare the robustness of three policies using the value of K in the convergence area. Fig. 6 shows how the robust ratio and regret computed by "MRR" change as the size of intervals (i.e., β) varies. Budget sizes are relative to the cost of removing all barriers. We see that as β increases, the robust ratio decreases and the regret increases almost linearly. "MRR" gives the largest robust ratio. Although "MRR" maximizes the robust ratio, it produces the smallest regret, implying that the two robustness metrics are correlated.

Finally, we test our algorithms on a large network of 9335 nodes, 7566 culverts and 596 dams with 5% budget. In this very difficult setting, we obtain results similar to those shown in Fig. 6 even without using the value of K in the convergence area. Due to the limitation of space, we do not show those similar figures here, but only visualize the computed policies in Fig. 7. The "midpoint" policy allocates most of the budget around the main stream, near the middle vertical line of the river. The adversarial policy can easily achieve much better value by taking actions in other important areas and assigns high probabilities if actions are taken (e.g., the adversarial policy) and low probabilities if actions are not taken (e.g., the decision policy.) In contrast, the "MRR" policy is more robust by allocating the budget to several important areas so that the adversarial policy cannot use the same trick to achieve much better value.

6 Conclusion

We describe an approximate robust optimization algorithm for a tree-structured stochastic network design problem, which is motivated by the river network design problem for fish conservation. The algorithm iteratively solves two optimization problem: the decision optimization problem and the ratio minimization problem. The former is encoded into



(b) "midpoint" (1.3×10^6) (c) "midpoint" adv. (2.7×10^6)

(d) "MRR" (1.7×10^6) (e) "MRR" adv. (2.2×10^6)

Figure 7: Visualization of four policies for β=0.3. Values shown in parenthesis. The accessibilities of edges are colored according to the top color bar. Dots represent removed (repaired) barriers. The adversarial midpoint and MRR policies are computed by RDP.

a MILP, and an FPTAS is developed for the latter, which is the harder problem. Empirically, we show that the policies computed by maximizing the robust ratio are more robust than policies computed by two other baseline methods. Besides finding policies of high robust ratio, our algorithm can also produce policies with small regret on large-scale networks. These algorithms provide new computational tools for environmental scientists who tackle decision problems with imprecise models.

Acknowledgments

This work was partially funded by a UMass Graduate School Dissertation Writing Fellowship awarded to the first author. Second author is supported by the research center at the School of Information Systems at the Singapore Management University.

References

Boutilier, C.; Patrascu, R.; Poupart, P.; and Schuurmans, D. 2003. Constraint-based optimization with the minimax decision criterion. In *International Conference on Principles and Practice of Constraint Programming*, 168–182. Springer.

Chen, W.; Lin, T.; Tan, Z.; Zhao, M.; and Zhou, X. 2016. Robust influence maximization. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 795–804. ACM.

Chen, W.; Wang, C.; and Wang, Y. 2010. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, 1029–1038. ACM.

He, X., and Kempe, D. 2014. Stability of influence maximization. *arXiv:1501.04579*.

Kempe, D.; Kleinberg, J.; and Tardos, E. 2003. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, 137–146.

Kouvelis, P., and Yu, G. 2013. *Robust discrete optimization and its applications*, volume 14. Springer Science & Business Media.

Kumar, A.; Singh, A. J.; Varakantham, P.; and Sheldon, D. 2016. Robust decision making for stochastic network design. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*.

Kumar, A.; Wu, X.; and Zilberstein, S. 2012. Lagrangian relaxation techniques for scalable spatial conservation planning. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence*, 309–315.

McGarigal, K.; Compton, B. W.; Jackson, S. D.; Plunkett, E.; Rolih, K.; Portante, T.; and Ene, E. 2011. Conservation assessment and prioritization system (CAPS). Technical Report November, Department of Environmental Conservation, Univ. of Massachusetts Amherst.

Neeson, T. M.; Ferris, M. C.; Diebel, M. W.; Doran, P. J.; OHanley, J. R.; and McIntyre, P. B. 2015. Enhancing ecosystem restoration efficiency through spatial and temporal coordination. *Proceedings of the National Academy of Sciences* 112(19):6236–6241.

O'Hanley, J. R., and Tomberlin, D. 2005. Optimizing the removal of small fish passage barriers. *Environmental Modeling and Assessment* 10(2):85–98.

Schichl, H., and Sellmann, M. 2015. Predisaster preparation of transportation networks. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, 709–715.

Sheldon, D.; Dilkina, B.; Elmachtoub, A.; Finseth, R.; Sabharwal, A.; Conrad, J.; Gomes, C.; Shmoys, D.; Allen, W.; Amundsen, O.; and Vaughan, W. 2010. Maximizing the spread of cascades using network design. In *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence*, 517–526.

Wu, X.; Sheldon, D.; and Zilberstein, S. 2014a. Rounded dynamic programming for tree-structured stochastic network design. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, 479–485.

Wu, X.; Sheldon, D.; and Zilberstein, S. 2014b. Stochastic network design in bidirected trees. In *Advances in Neural Information Processing Systems*, 882–890.

Wu, X.; Sheldon, D.; and Zilberstein, S. 2016. Optimizing resilience in large scale networks. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*.

Appendix

Lemma 1. *There exists an optimal policy-parameter pair $(\pi'^*, \boldsymbol{p}^*)$ for Problem (4) with the following property. Suppose π'^* takes action i and the decision policy π takes action j on edge e. If $j \neq i$, then $p^*_{e|i} = \overline{p}_{e|i}$ and $p^*_{e|j} = \underline{p}_{e|j}$. Otherwise, $p^*_{e|i}$ is either $\underline{p}_{e|i}$ or $\overline{p}_{e|i}$.*

Proof. Suppose π'^* takes action i and the decision policy π takes action j on edge e. Let us first consider the case when $j \neq i$. Since only π takes action j on e, the term $p_{e|j}$ only appears in the numerator of the robust ratio in (4), and $p_{e|i}$ only appears in the denominator. Then, the minimization w.r.t. $\boldsymbol{p}$ in (4) can be written as

$$\min_{\boldsymbol{p}} \frac{ap_{e|j} + b}{cp_{e|i} + d}$$

where a, b, c, d are constants w.r.t. $p_{e|i}$ and $p_{e|j}$. Since all rewards and probabilities are nonnegative, coefficients a and c are nonnegative. The optimal probability setting will satisfy $p^*_{e|i} = \overline{p}_{e|i}$ and $p^*_{e|j} = \underline{p}_{e|j}$, which proves the first part of the lemma.

Let us consider the case when $j = i$. Now, $p_{e|i}$ will appear in both numerator and denominator. In this case, we have

$$\begin{aligned}
&\min_{\boldsymbol{p}} \frac{ap_{e|i} + b}{cp_{e|i} + d} \\
=&\min_{\boldsymbol{p}} \frac{\frac{a}{c}(cp_{e|i} + d) + b - \frac{ad}{c}}{cp_{e|i} + d} \\
=&\min_{\boldsymbol{p}} \frac{a}{c} + \frac{b - \frac{ad}{c}}{cp_{e|i} + d}
\end{aligned}$$

where a, b, c, d are nonnegative constants w.r.t. $p_{e|i}$. If $b \geq \frac{ad}{c}$, the optimal probability setting will set $p_{e|i} = \overline{p}_{e|i}$. Otherwise, it will set $p_{e|i} = \underline{p}_{e|i}$. In summary, $p^*_{e|i}$ is either the upperbound or the lowerbound, which proves the second part. □

Corollary 1. *For a fixed π, only $|A_e| + 1$ actions in A^s_e are needed to compute $(\pi'^*, \boldsymbol{p}^*)$.*

Proof. Due to Lemma 1, if π' and π take different actions, there is only one possible probability setting that we need to consider. If they take the same action (say i), there are two cases $p_{e|i} = \underline{p}_{e|i}$ or $p_{e|i} = \overline{p}_{e|i}$ while the probabilities of other actions than i can be chosen arbitrarily and don't affect the objective value of both the decision policy and the adversarial policy. □

Theorem 4. $T(n_u) = O(\frac{n_u^4}{\mu^2})$.

Proof. We have

$$\begin{aligned}
T(n_u) &= O(m_v^a m_v^d m_w^a m_w^d) + T(n_v) + T(n_w) \\
&\leq c\frac{n_v^2 n_w^2}{\mu^2} + T(n_v) + T(n_w) \\
&\leq \max_{0 \leq k \leq n_u - 1} c\frac{k^2(n_u - k - 1)^2}{\mu^2} + T(k) + T(n_u - k - 1)
\end{aligned}$$

where n_u, n_v, n_w are the numbers of nodes in subtree at u, v, w and $n_u = n_v + n_w + 1$.

To show that $T(n_u) = O(\frac{n_u^4}{\mu^2})$, we use induction. For the base case, the DP table at a leave node has only one tuple, so $T(1) = O(1) = O(\frac{1}{\mu^2})$ as $\mu < 1$. To do the induction, let v and w be the two children of u and assume that $T(n_v) = c_1\frac{n_v^4}{\mu^2}$ and $T(n_w) = c_2\frac{n_w^4}{\mu^2}$. Let $c' = \max\{c_1, c_2, c\}$ where c is the constant in previous inequalities. Continuing the derivation of $T(n_u)$, we have

$$\begin{aligned}
T(n_u) &\leq \frac{c'}{\mu^2} \max_{0 \leq k \leq n_u - 1} 2k^2(n_u - k - 1)^2 + k^4 + (n_u - k - 1)^4 \\
&\leq \frac{c'}{\mu^2} \max_{0 \leq k \leq n_u - 1} (k^2 + (n_u - k - 1)^2)^2 \\
&\leq \frac{c'}{\mu^2} \max_{0 \leq k \leq n_u - 1} (k^2 + 2k(n_u - k - 1) + (n_u - k - 1)^2)^2 \\
&\leq \frac{c' n_u^4}{\mu^2}
\end{aligned}$$

Thus, we have shown that $T(n_u) = O(\frac{n_u^4}{\mu^2})$. □