2026 The International Mathematical Modeling Challenge (IMMC)

Team Control Number: **IMMC2026033**

Abstract

Protecting vast wildlife reserves like Etosha National Park (approximately 22,935 km^2) with severely constrained resources demands a mathematically rigorous strategy that abandons the flawed pursuit of total coverage. This work presents a comprehensive, grid-based operations research framework to optimize wildlife protection by strategically executing spatial triage against competing ecological threats.

For **Problem 1**, we identify and prioritize three major conservation challenges—**poaching**, **wildfires**, and **boundary breaches**—using a risk matrix. We translate the abstract concept of **"protection"** into a quantitative **Spatial Triad** evaluated over a high-resolution grid. This includes a **point-based Anti-Poaching Index (API)** maximizing joint detection and interception probabilities, an **area-based Habitat Protection Index (HPI)** minimizing the miss-probability of overlapping sensor networks, and a **line-based Boundary Integrity Index (BII)** capturing fence security via cyclic volunteer patrols.

In **Problem 2**, we formulate a **constrained non-linear integer programming model** to maximize the risk-weighted composite protection score ($\mathcal{F}$). The model endogenously prioritizes high-risk zones using **biome-specific** geographic modifiers (e.g., terrain impedance, proximity to roads). To efficiently navigate the massive combinatorial space, we developed a custom **Cost-Benefit Greedy Algorithm** driven by marginal gain-to-cost ratios ($\Delta\mathcal{F}/\text{Cost}$). Our optimization precisely allocates the 300-unit budget to deploy **39 ranger squads (195 personnel), 9 UAVs, 59 AI cameras, and 15 volunteer teams (90 individuals)**, achieving a $\mathcal{F}^*$ score of **0.787**. Spatially, this deployment validates our **triage strategy**: rangers and cameras are heavily clustered around critical road networks and waterholes, while UAVs sweep expansive forest and savanna regions, and volunteers secure the perimeter.

Problem 3 extends the framework to analyze protection across four orthogonal spatio-temporal scenarios (**seasonal and diurnal cycles**). Forward evaluation reveals a degradation in the baseline protection score (**dropping from 0.787 in dry-days to 0.565, 0.612, and 0.560** in alternative scenarios), exposing critical vulnerabilities during **wet-season nights**. Utilizing **Inverse Optimization** coupled with a **binary search** heuristic, we calculate the minimum active personnel required to maintain a protection baseline of $\mathcal{F}^* = 0.7$. The results highlight the operational strain imposed by adverse environmental conditions: to maintain this baseline standard during the worst-case **Wet-Night scenario**, the park requires a **workforce multiplier of** $1.92\times$ compared to the standard Dry-Day allocation.

For **Problem 4**, **sensitivity analysis** utilizing elasticity indices and **Tornado charts** confirms the model's overarching systemic resilience. We demonstrate that under extreme environmental stress or staffing shortages, capital-intensive technology behaves as a highly elastic substitute for human labor mobility. In **Problem 6**, we demonstrate the framework's universal applicability by adapting it to **Lorentz National Park (Indonesia)** and **Yellowstone (USA)**, recalibrating core matrices such as 3D terrain impedance and aerial efficacy to different ecosystems.

Finally, we evaluate the model's **strengths** and **limitations**, synthesizing our findings into an actionable, non-technical **letter** for the IMMC. In conclusion, our model equips conservation organizations with a highly scalable tool to quantify, optimize, and execute wildlife protection strategies.

Keywords: Wildlife Conservation, Spatial Triad Model, Joint Probability Framework, Greedy Optimization, Spatio-Temporal Allocation, Etosha National Park

Contents

1 Introduction

1.1 Background

National parks and large wildlife reserves play a critical role in protecting global biodiversity, yet their vast size and complex terrain make continuous monitoring difficult. **Etosha National Park in Namibia** exemplifies this challenge [1]. Spanning approximately 22,935 km^2, the park features a roughly 4,800 km^2 central salt pan [2] and a network of natural and man-made waterholes. These waterholes concentrate wildlife-including the endangered black rhinoceros-making them predictable targets for poaching. The park must also manage an approximately 850 km perimeter fence to prevent unauthorized entry and human-wildlife conflict, while monitoring vast open savannas for wildfires [2, 3].

Etosha currently deploys approximately 295 Ministry personnel to protect this ecosystem [2], yet limited staff and resources make it impossible to cover such a vast area comprehensively. Traditional patrols alone are insufficient, motivating the integration of modern conservation technologies-including Unmanned Aerial Vehicle (UAV), fixed monitoring systems, and AI-assisted detection (Figure 1.1). The core challenge for the International Mission for Monitoring Conservation is to quantify "protection" as a mathematically measurable outcome, enabling conservation organizations to strategically allocate constrained human and technological resources against competing threats.

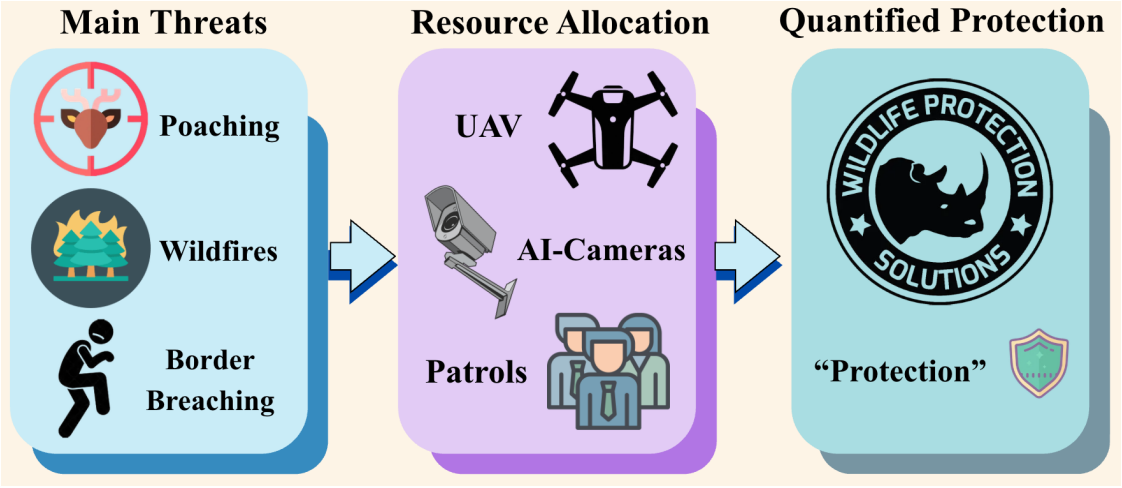


Figure 1.1: Main threats with heterogeneous resources for quantified wildlife protection.

1.2 Problem Restatement

The International Mission for Monitoring Conservation (IMMC) requires a mathematical framework to optimize wildlife protection in Etosha National Park, bridging the gap between its massive scale and limited resources. Our team addresses the following core objectives:

1. **Threat Identification and Definition of "Protection":** Identify and prioritize the major conservation threats in the park, then define "protection" in measurable terms that can vary by species or region and integrate into a quantitative modeling framework.
2. **Resource Allocation Modeling:** Develop a mathematical model to distribute conservation resources-personnel, patrols, and technology-across identified threat priorities, accounting for resource scarcity, geographic constraints, and environmental uncertainty.
3. **Temporal Analysis and Staffing:** Evaluate protection effectiveness over time, with emphasis on operational feasibility and practical resource utility.
4. **Sensitivity and Scenario Analysis:** Assess how changes in key assumptions or resource availability-such as staffing cuts, reduced technology, or altered patrol schedules-affect overall protection efficacy.
5. **Model Strengths and Limitations:** Critically examine the advantages and constraints of our approach, including assumptions that may limit real-world applicability.
6. **Global Generalization:** Demonstrate the framework's adaptability by applying it to two additional national parks or wildlife reserves on different continents.
7. **Letter to the IMMC:** Synthesize main findings into a non-technical letter to IMMC.

1.3 Our Work

We developed a comprehensive, grid-based operations research model to guide protection strategies for large wildlife reserves, utilizing Etosha National Park as a foundational case study. Our work is structured into four main analytical phases:

1. Conservation Priorities and Defining "Protection" (Problem 1). Using a risk matrix, we systematically prioritized Etosha's conservation challenges: poaching (highest priority), wildfires, and boundary breaches. We translated the abstract concept of "protection" into a quantitative **Spatial Triad**: point-based (Anti-Poaching Index, maximizing joint detection and interception probabilities), area-based (Habitat Protection Index, minimizing the miss-probability of overlapping sensor networks), and line-based (Boundary Integrity Index, securing perimeters via cyclic volunteer patrols).

2. Resource Allocation Model (Problem 2). We formulated a constrained non-linear optimization model to maximize the composite protection score $\mathcal{F}$ (weighted 20:12:9). By discretizing the 22,935 km^2 park into a high-resolution grid, the model strategically allocates rangers, UAVs, AI cameras, and volunteers. It integrates biome-specific geographic modifiers (e.g., roads, vegetation density) to endogenously force resource concentration on high-risk zones. To efficiently navigate this massive combinatorial space, we developed a **Cost-Benefit Greedy Algorithm** that iteratively deploys assets based on their maximum marginal protection gain per unit cost ($\Delta\mathcal{F}/\text{Cost}$).

3. Protection Over Time (Problem 3). We extended our static model by introducing orthogonal temporal dimensions: **seasonal (dry vs. wet) and diurnal (day vs. night)**. Forward evaluation revealed critical protection gaps, particularly during wet-season nights. Using **inverse optimization**, we calculated the minimum active personnel required to maintain a strict protection baseline. This culminated in our recommendation for a highly efficient, **Tiered Dynamic Deployment** strategy.

4. Sensitivity and Adaptation (Problems 4 and 6). Through elasticity indices and extreme scenario stress tests, we validated the model's structural resilience, mathematically demonstrating how capital-intensive technology acts as an elastic substitute for human labor during severe staffing crunches. Finally, we proved the model's global transferability by mapping its core mathematical matrices (terrain

impedance, detection efficacy R_{uav}) to entirely different ecosystems: the dense, mountainous tropical canopies of Lorentz National Park (Indonesia) and the Yellowstone (USA).

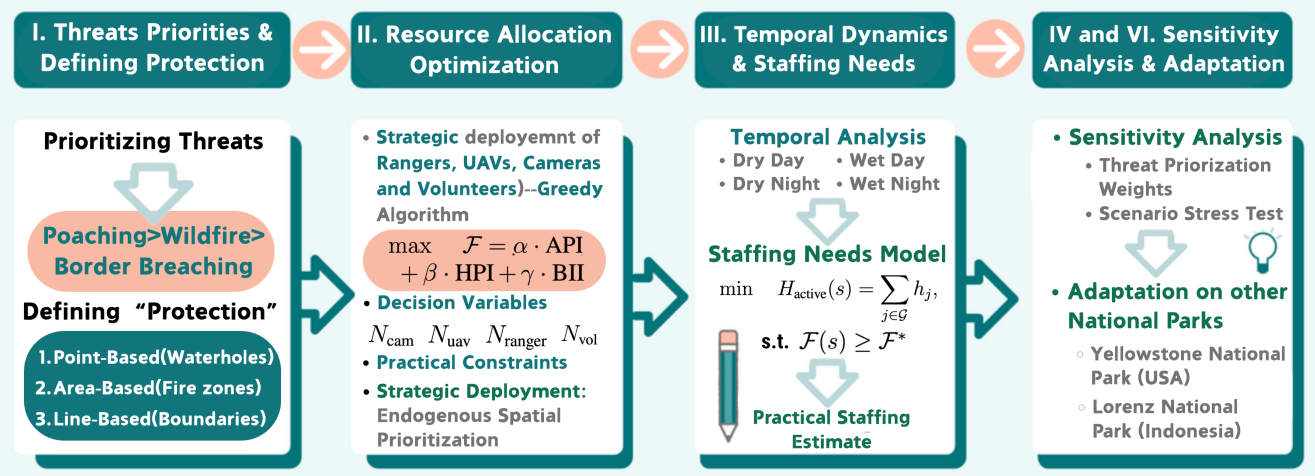


Figure 1.2: The holistic workflow of our modeling and optimization approach.

2 Assumptions and Justifications

To ensure the tractability of our mathematical model while preserving the complex operational realities of Etosha National Park, we make the following core assumptions:

- **Assumption 1: Geographic and Infrastructure Stability.** We assume that the park's primary topographic features, road networks, waterholes, and main gates remain geographically fixed throughout the modeling period.
  Justification: While animal migrations are dynamic, the physical infrastructure (roads, fences) and critical ecological nodes (waterholes) are stationary. Using a static grid system to represent these fixed spatial elements provides a stable mathematical foundation for evaluating relative threat probabilities and calculating travel distances.
- **Assumption 2: Independent and Redundant Detection Sources.** We assume that different surveillance mechanisms (e.g., human patrols, UAVs, AI cameras, and satellites) operate as statistically independent detection events.
  Justification: In field operations, a UAV flying overhead and a camera mounted on a tree capture different perspectives without interfering with one another. This independence allows us to mathematically calculate the overarching joint detection probability using the standard formula $P_{\text{total}} = 1 - \prod(1 - p_i)$, logically reflecting that redundant overlapping coverage minimizes the chance of complete omission.
- **Assumption 3: Diminishing Marginal Returns on Localized Resources.** We assume that the protection efficacy (e.g., detection probability) within a specific spatial grid exhibits asymptotic saturation as more identical resources are continuously stacked in that same location.
  Justification: This reflects fundamental operational realities. For instance, while deploying the first AI camera at a waterhole dramatically reduces detection time, adding a fourth or fifth camera to the same specific location provides negligible marginal benefit due to overlapping fields of view. This assumption naturally drives our optimization algorithm to seek sparse, distributed solutions rather than hoarding all assets in a single cell.
- **Assumption 4: Independence of Threat Modalities.** We assume that poaching incidents, natural wildfires, and boundary fence breaches occur as statistically independent events.
  Justification: While complex real-world correlations might occasionally exist (e.g., a poacher accidentally starting a fire to create a distraction), explicitly modeling these cross-threat dependencies requires unavailable high-order interaction data. Assuming independence allows us to mathematically formulate the global objective function ($\mathcal{F}$) as a linear, risk-weighted sum of the three distinct

protection indices without requiring complex covariance penalties.

- **Assumption 5: Spatial Influence via Grid Neighborhoods.** We assume that mobile assets stationed in a specific grid cell exert a localized sphere of influence (e.g., a 3×3 grid for rangers, and a 5×5 grid for UAVs) with an orthogonal and diagonal distance decay.
  Justification: Rangers do not teleport; they establish semi-permanent forward operating bases (fly camps) and patrol the surrounding vicinity. Modeling their influence as a decaying multi-grid neighborhood accurately captures their territorial patrol patterns and finite mobility without requiring complex continuous-time routing algorithms.
- **Assumption 6: Existing Baseline Infrastructure Operates Passively.** We assume that the pre-existing main camps and lookout towers are permanently staffed to maintain basic park operations, providing a constant, baseline surveillance probability without consuming our optimization budget.
  Justification: A park of Etosha's magnitude relies on inherent administrative infrastructure to function. By treating these nodes as passive, zero-marginal-cost baseline coverage, our model can exclusively focus the constrained budget on deploying *additional,* tactical resources to critical vulnerabilities.

3 Variables and Notations

The primary symbols and variables utilized throughout this report are outlined in Table 3.1. Any additional, highly specific, or localized mathematical symbols will be introduced and explicitly defined at their point of usage.

Table 3.1: **Key Mathematical Variables and Notations**

Category	Symbol	Description
Sets & Indices	$\mathcal{G}$	The set of all discrete 1 km $\times$ 1 km grid cells comprising the park
	(r, c)	The spatial coordinate index (row, column) for a specific grid cell
	k	Index for boundary fence segments ($k \in \mathcal{S}$)
	s	Temporal scenario index (e.g., Dry-Day, Dry-Night, Wet-Day, Wet-Night)
Decision Variables	$N_{\text{ranger}}(r, c)$	Number of mobile ranger squads deployed at grid (r, c)
	$N_{\text{uav}}(r, c)$	Number of UAVs allocated to grid (r, c) for aerial sweeps
	$N_{\text{cam}}(r, c)$	Number of fixed AI cameras installed at grid (r, c) (max 3 per waterhole)
	N_{vol}	Total number of volunteers deployed for cyclic boundary patrols
Protection Metrics	$\mathcal{F}$	Global Comprehensive Protection Score (Objective Function)
	API, HPI, BII	Aggregate sub-indices: Anti-Poaching, Habitat Protection, Boundary Integrity
	$V_{\text{API}}(r, c)$	The point-level poaching protection value computed for grid (r, c)
	$P_{\text{miss}}(r, c)$	The joint probability of a wildfire going undetected at grid (r, c)
	I_k	The physical intactness probability of boundary segment k
Detection & Strength	$S_{\text{ranger}}, S_{\text{uav}}$	Distance-weighted patrol and aerial surveillance strength at a given grid
	P_{detect}	The joint probability of successfully detecting a poaching threat
	$P_{\text{patrol}}, P_{\text{camera}}$	Independent threat detection probabilities of specific resource classes
	k_{patrol}	Saturation coefficient converting patrol strength into probability
	κ	Efficiency saturation multiplier for stacked AI cameras
Risks & Weights	α, β, γ	Global priority weights assigned to poaching, wildfire, and boundary threats
	$R_{\text{poach}}(r, c)$	Localized inherent poaching risk weight for grid (r, c)
	$R_{\text{fire}}(r, c)$	Localized inherent wildfire risk weight for grid (r, c)
	w_k	Normalized vulnerability weight of boundary segment k

4 Problem 1: Conservation Priorities and Defining Protection

4.1 Identifying and Prioritizing Conservation Challenges

Etosha National Park faces multiple heterogeneous threats that vary significantly by species, habitat, location, and driving factors (human-driven vs. natural) [2]. Based on the provided background data and geographical characteristics, we identify three primary conservation challenges.

Threat Classification and Risk Matrix. To systematically evaluate these challenges and lay the groundwork for our resource allocation objective function, we developed a risk stratification framework. We assess each threat k along two dimensions: **Severity** (S_k, the potential for irreversible damage to endangered species or core ecosystems) and **Likelihood** (L_k, the probability of occurrence based on historical pressure and spatial accessibility).

The composite risk $\mathcal{R}_k$ for each threat category $k \in \{\text{Poaching}, \text{Wildfire}, \text{Breach}\}$ is formulated as:

$$\mathcal{R}_k = S_k \times L_k, \tag{4.1}$$

where both S_k and L_k are quantified on a discrete scale of 1 to 5. The resulting prioritization is summarized in Table 4.1.

Table 4.1: Risk Matrix for Etosha National Park Conservation Challenges [2]

Threat Type (k)	**Primary Target & Spatial Feature**	**Severity** (S_k)	**Likelihood** (L_k)	**Risk Score** ($\mathcal{R}_k$)	**Priority Level**
Poaching	Black rhinoceros (Endangered, Point-based)	5	4	20	**High**
Wildfires	Savanna/grassland (Area-based)	3	4	12	**Medium-High**
Boundary Breaches	Human-wildlife conflict (Line-based)	3	3	9	**Medium-Low**

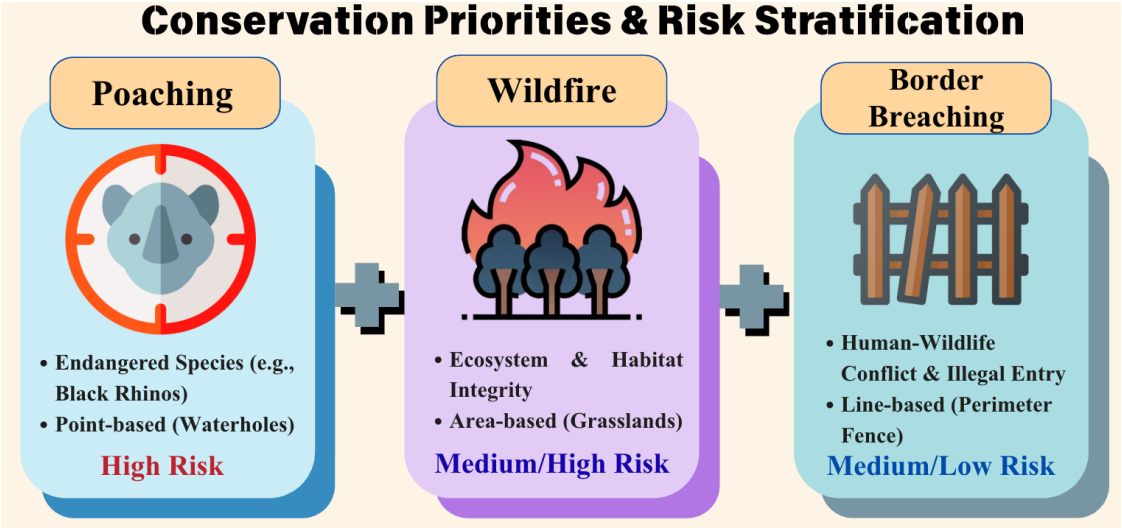


Figure 4.1: Risk stratification mapping the spatial dimensions (Point, Area, Line) of Etosha's primary conservation priorities.

High Priority: Point-Based Poaching of Endangered Species. Black rhinoceroses represent Etosha's most vulnerable species, classified as endangered and facing significant, documented poaching pressure [4]. While animals roam across the landscape, poaching is fundamentally a point-based problem - poachers often target specific high-value locations where prey predictably congregates. The park's waterholes serve as critical wildlife aggregation sites, creating highly localized, predictable targets. Furthermore, proximity to the 3,551 km road network facilitates rapid access and extraction, concentrating risk along accessible corridors. Although the threat manifests across space, effective protection requires deploying discrete assets at strategic points, including ranger patrols and camera

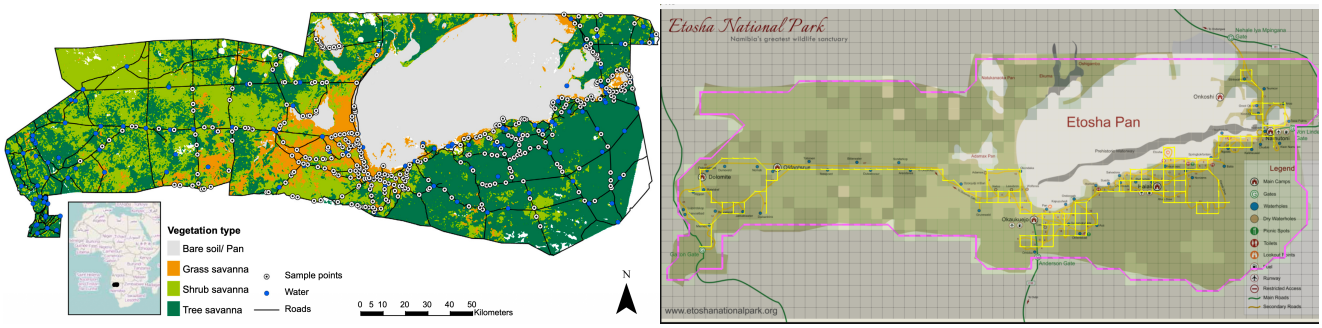
traps. This point-based deployment strategy enables rapid-response intervention before poachers can complete their kill and escape.

Medium-High Priority: Area-Based Habitat Disruptions (Wildfires). Recent years have witnessed wildfires impacting large portions of the park. The main terrain types, specifically the savanna and grassland ecosystems, are highly combustible and susceptible to rapid fire spread. Unlike targeted poaching, wildfire risk is **area-based**, threatening habitat integrity across expansive, continuous zones. While the approximately 4,800 km^2 central salt pan acts as a natural firebreak, the vast surrounding vegetation remains highly vulnerable and requires broad spatio-temporal monitoring.

Medium-Low Priority: Line-Based Boundary Breaches. The park's boundary relies on an approximately 850 km perimeter fence. This boundary faces constant pressure from both immense wildlife populations-such as the estimated 2,500 elephants in the broader ecosystem-and unauthorized human activities. Breaches compromise park security, enabling illegal entry and severe human-wildlife conflict outside the reserve [2]. This threat is **line-based**, distributed continuously along the perimeter, necessitating integrity maintenance over geographic distance.

4.2 Defining Point-, Area-, and Line-Based Protection Metrics

Protection within Etosha's vast 22,935 km^2 expanse cannot be adequately characterized as a simple binary state (i.e., protected vs. unprotected). Instead, we propose a multi-dimensional, quantitative framework in which **"protection" is formally defined as the probability of successful threat detection and interception**. This framework is operationalized through three complementary, spatially-resolved metrics computed over a discrete grid $\mathcal{G}$ partitioning the park into cells indexed by (r, c) illustrated in Figure 4.2.



(a) Etosha Vegetation Landform (b) Etosha Map Discretization

Figure 4.2: Spatial discretization of Etosha National Park.

Point-Based Protection: Anti-Poaching Index (API)

Effective anti-poaching protection requires both the *detection* of an incursion and the *physical interception* of the threat by rangers. We first quantify the effective resource strength available at grid cell (r, c) by aggregating contributions from neighboring cells via distance-decay weighting. Specifically, the ranger and UAV strength fields are defined as:

$$S_{\text{ranger}}(r, c) = \sum_{(dr,\, dc) \in \mathcal{N}_3} w_{dr,dc} \cdot N_{\text{ranger}}(r + dr,\ c + dc) + S_{\text{camp}}(r, c), \tag{4.2}$$

$$S_{\text{uav}}(r, c) = \sum_{(dr,\, dc) \in \mathcal{N}_5} w'_{dr,dc} \cdot N_{\text{uav}}(r + dr,\ c + dc), \tag{4.3}$$

where $N_{\text{ranger}}(\cdot)$ and $N_{\text{uav}}(\cdot)$ denote the number of ranger units and UAVs deployed at a given cell, respectively; $\mathcal{N}_3$ and $\mathcal{N}_5$ denote the 3×3 and 5×5 Moore neighborhoods centered at (r, c); $w_{dr,dc}$

and $w'_{dr,dc}$ are the corresponding distance-decay weights; and $S_{\text{camp}}(r,c)$ is the passive surveillance contribution from nearby ranger camps.

The marginal detection probabilities of the three independent monitoring modalities—ground patrols, UAVs, and AI-enabled camera traps—are modeled as:

$$P_{\text{patrol}}(r,c) = 1-e^{-k_{\text{patrol}}\cdot S_{\text{ranger}}(r,c)}, \quad P_{\text{uav}}(r,c) = 1-e^{-k_{\text{patrol}}\cdot S_{\text{uav}}(r,c)}, \quad P_{\text{camera}}(r,c) = 1-\frac{1}{1+\kappa\cdot N_{\text{cam}}(r,c)}, \tag{4.4}$$

where $k_{\text{patrol}} > 0$ is a sensitivity scaling constant governing the effectiveness of mobile assets, $\kappa > 0$ is the analogous sensitivity parameter for camera traps, and $N_{\text{cam}}(r,c)$ is the number of AI cameras deployed in cell (r,c). Treating the three systems as statistically independent sensors, the composite detection probability is given by the complement of the joint miss probability:

$$P_{\text{detect}}(r,c) = 1 - \big(1-P_{\text{patrol}}\big)\big(1-P_{\text{uav}}\big)\big(1-P_{\text{camera}}\big). \tag{4.5}$$

Since detection alone is insufficient without ranger capacity to physically respond, the cell-level anti-poaching protection value couples detection probability with interception capacity:

$$V_{\text{API}}(r,c) = P_{\text{detect}}(r,c)\cdot\Big(1-e^{-S_{\text{ranger}}(r,c)}\Big). \tag{4.6}$$

The **park-wide Anti-Poaching Index (API)** is then defined as the poaching-risk-weighted average of V_{API} across all grid cells:

$$\text{API} = \frac{\displaystyle\sum_{(r,c)\,\in\,\mathcal{G}} R_{\text{poach}}(r,c)\cdot V_{\text{API}}(r,c)}{\displaystyle\sum_{(r,c)\,\in\,\mathcal{G}} R_{\text{poach}}(r,c)}, \tag{4.7}$$

where $R_{\text{poach}}(r,c) \geq 0$ is the inherent poaching risk weight of cell (r,c), derived from habitat characteristics and historical incident data.

Area-Based Protection: Habitat Protection Index (HPI)

Wildfires threaten expansive contiguous habitats and demand continuous, multi-layered surveillance. For each grid cell (r,c), we define the probability that a fire onset goes *completely undetected* as the joint probability of simultaneous failure across all active monitoring systems:

$$P_{\text{miss}}(r,c) = \big(1-P_{\text{uav}}\big)\big(1-P_{\text{ranger}}\big)\big(1-P_{\text{sat}}\big)\big(1-P_{\text{lookout}}\big)\big(1-P_{\text{camp}}\big), \tag{4.8}$$

where P_{sat} is a spatially uniform baseline detection probability from satellite imagery, and P_{lookout} and P_{camp} represent the static passive detection coverage provided by observation towers and ranger camps, respectively. The **Habitat Protection Index (HPI)** is defined as the fire-risk-weighted average of the successful detection probability across the park:

$$\text{HPI} = \frac{\displaystyle\sum_{(r,c)\,\in\,\mathcal{G}} R_{\text{fire}}(r,c)\cdot\big(1-P_{\text{miss}}(r,c)\big)}{\displaystyle\sum_{(r,c)\,\in\,\mathcal{G}} R_{\text{fire}}(r,c)}, \tag{4.9}$$

where $R_{\text{fire}}(r,c) \geq 0$ denotes the fire-risk weight of cell (r,c), informed by vegetation type, fuel load, and seasonal aridity.

Line-Based Protection: Boundary Integrity Index (BII)

The park's approximately 850 km perimeter fence is monitored by volunteer patrol groups. Given a total of N_{vol} deployed volunteers organized into groups of six, the number of patrol groups is $G = \lfloor N_{\text{vol}}/6 \rfloor$. The expected time required to discover a fence breach is determined by the patrol cycle over the total

fence length L_{total}:

$$T_{\text{find}} = \frac{L_{\text{total}}}{G \times V_{\text{patrol}} \times T_{\text{patrol}}} \times 24 \quad \text{(hours)}, \tag{4.10}$$

where V_{patrol} is the average patrol speed (km/h) and T_{patrol} is the daily active patrol duration (hours/day). Once discovered, the fence must be repaired; the total duration during which a breach remains exploitable is thus $T_{\text{broken}} = T_{\text{find}} + T_{\text{fix}}$, where T_{fix} is the mean repair time. For fence segment k characterized by a historical damage frequency λ_k (breaches per unit time), the *intactness score* I_k represents the expected fraction of the total simulation period T_{total} during which the segment remains intact:

$$I_k = \max\left(0,\ 1 - \frac{\lambda_k \cdot \left(T_{\text{find}} + T_{\text{fix}}\right)}{T_{\text{total}}}\right) \ \in\ [0,\ 1]. \tag{4.11}$$

The overall **Boundary Integrity Index (BII)** is computed as the weighted average of intactness scores across all perimeter segments $\mathcal{S}$:

$$\text{BII} = \sum_{k \in \mathcal{S}} w_k \cdot I_k = \frac{\sum_{k \in \mathcal{S}} w_k \max\left(0,\ 1 - \frac{\lambda_k \cdot \left(T_{\text{find}} + T_{\text{fix}}\right)}{T_{\text{total}}}\right)}{\sum_k w_k} \tag{4.12}$$

where $w_k \geq 0$ are segment-level importance weights satisfying $\sum_k w_k = 1$, reflecting the ecological and operational significance of each boundary section.

Integrated Protection Framework

As summarized in Table 4.2 and Figure 4.3, the three indices—API, HPI, and BII—constitute a **spatial triad protection system** that unifies point-, area-, and line-based threat defense within a cohesive, grid-resolved mathematical framework. Together, they enable a comprehensive, spatially explicit quantification of protection efficacy across Etosha's heterogeneous landscape.

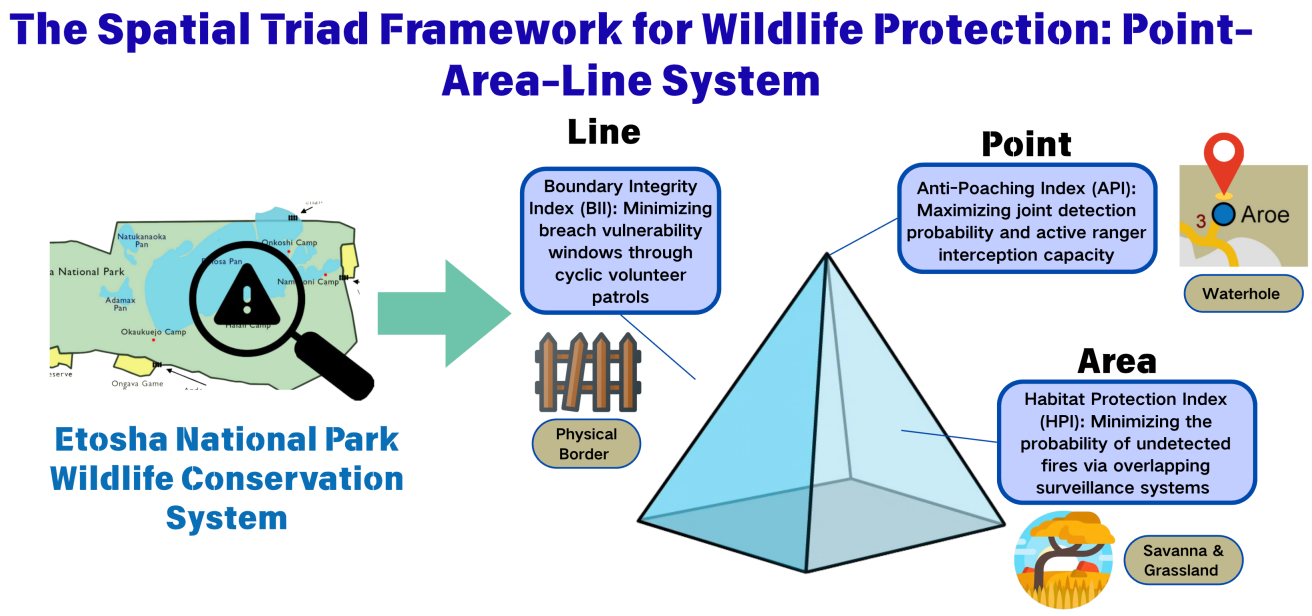


Figure 4.3: The Spatial Triad Framework for Wildlife Protection: Point-Area-Line System.

5 Problem 2: Resource Allocation Model

Building upon the Point-Area-Line metrics defined in Problem 1, we formulate a mathematical model to allocate limited protection resources across Etosha National Park. Our model explicitly incorporates **geographic considerations**, **uncertainty**, and the imperative for **strategic deployment** rather than complete coverage.

Table 4.2: Summary of the Spatial Triad Protection Metrics.

Metric	Symbol	Spatial Nature	Primary Threat	Spatial Target
Anti-Poaching Index	API	Point-focused	Poaching	Cells with elevated poaching risk (waterholes, etc.)
Habitat Protection Index	HPI	Area-based	Wildfires	Vulnerable biomes across $\mathcal{G}$
Boundary Integrity Index	BII	Line-based	Fence breaches	Perimeter segments $k \in \mathcal{S}$

5.1 Strategic Deployment via Endogenous Spatial Prioritization

Rather than imposing uniform security coverage or manual exclusion rules, our model achieves strategic resource deployment *endogenously*. By embedding inherent risk weights—R_{poach}, R_{fire}, and w_k—directly into the grid evaluation, the optimization algorithm is compelled to channel constrained resources toward locations where the marginal return on protection, measured as $\Delta\mathcal{F}/\text{Cost}$, is maximized.

Biome Classification and Geographic Modifiers. To systematically characterize the park's heterogeneous terrain, we discretize the study area into a high-resolution grid $\mathcal{G}$ composed of $1\,\text{km} \times 1\,\text{km}$ cells. Each cell (r, c) is assigned baseline values for travel resistance, fire risk, and poaching risk derived from its underlying biome type. For instance, Tree Savanna carries a relatively high fire risk of 1.0, whereas the barren Etosha Pan is assigned a value of 0.0.

These baseline values are further modulated by **geographic modifiers** that reflect the influence of physical infrastructure:

- **Waterholes:** Increase poaching risk by $+0.8$, as they serve as primary attractants for endangered wildlife and therefore concentrate poaching activity. Also decreases fire risk by -0.2.
- **Dry waterholes:** Increase fire risk by $+0.4$; while no longer holding water, these sites often support dense vegetation that desiccates in the dry season, creating a concentrated fuel load more prone to ignition and rapid fire spread.

This multi-layered spatial logic naturally directs resources away from the low-risk salt pans and concentrates them around road-accessible waterholes and high-density savannas.

5.2 Deployable Assets and Decision Variables

Our framework operationalizes protection through four distinct, deployable asset types, each characterized by specific operational capabilities, spatial ranges, and associated costs, as detailed in Table 5.1.

Table 5.1: Decision Variables for Resource Allocation.

Variable	Domain	Physical Interpretation
$N_{\text{cam}}(r, c)$	$\mathbb{Z}_{\geq 0},\ \max = 3$	Number of fixed AI camera traps deployed at cell (r, c)
$N_{\text{uav}}(r, c)$	$\mathbb{Z}_{\geq 0}$	Number of UAVs stationed at cell (r, c) for aerial surveillance
$N_{\text{ranger}}(r, c)$	$\mathbb{Z}_{\geq 0}$	Number of ranger patrol units assigned to cell (r, c) for detection and interception
N_{vol}	$\mathbb{Z}_{\geq 0}$	Total number of volunteers dedicated to perimeter fence patrol

5.3 Objective Function Formulation

The optimization objective is to maximize the holistic protection of the park, defined as the priority-weighted sum of the three spatial protection indices introduced in Section 4.2:

$$\max_{\{N_{\text{cam}},\, N_{\text{uav}},\, N_{\text{ranger}},\, N_{\text{vol}}\}} \quad \mathcal{F} = \underbrace{\alpha \cdot \text{API}}_{\text{Anti-poaching term (point)}} + \underbrace{\beta \cdot \text{HPI}}_{\text{Wildfire monitoring term (area)}} + \underbrace{\gamma \cdot \text{BII}}_{\text{Boundary control term (line)}} . \tag{5.1}$$

The priority weights α, β, and γ are derived from the Risk Matrix analysis in Problem 1 using a relative importance ratio of $20:12:9$, yielding $\alpha \approx 0.488$, $\beta \approx 0.293$, and $\gamma \approx 0.220$.

5.4 Constraints and Complete Optimization Formulation

To ensure practical feasibility, the allocation strategy is subject to a set of constraints governing budgetary limits, asset inventories, and deployment rules.

- **Global Budget Constraint:** The total weighted deployment cost across all assets must not exceed the budget ceiling $R_{\max} = 300$ (budget units). Unit costs are calibrated to reflect relative capital and operational expenditures: UAVs (7.0), ranger patrols (5.0), volunteers (2.0), and AI cameras (0.2).
- **Asset Inventory Limits:** Given the finite availability of specialized equipment and personnel, the total UAV fleet is capped at 16 units and the volunteer pool at 180 individuals.
- **Camera Saturation Limit:** To prevent redundant field-of-view overlap at a single location, at most 3 camera units may be installed within any individual cell.

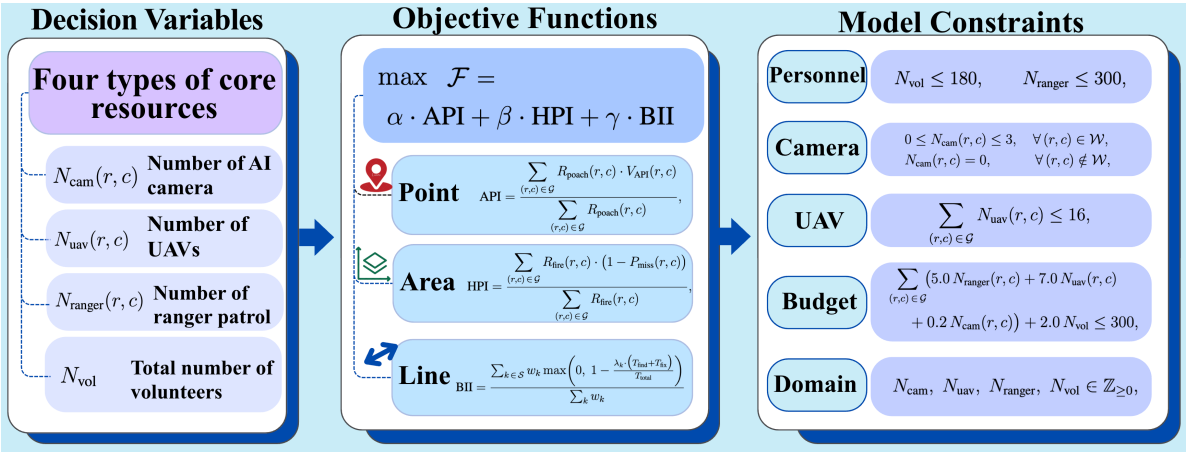


Figure 5.1: Structural mapping of the complete constrained resource allocation model.

Consolidating the objective function (5.1) and all operational constraints, the complete formulation of the integer programming problem is:

$$\max \quad \mathcal{F} = \alpha \cdot \text{API} + \beta \cdot \text{HPI} + \gamma \cdot \text{BII}$$

$$\text{s.t.} \quad \begin{cases} \displaystyle\sum_{(r,c)\in\mathcal{G}} \left(5.0\,N_{\text{ranger}}(r,c) + 7.0\,N_{\text{uav}}(r,c) + 0.2\,N_{\text{cam}}(r,c)\right) + 2.0\,N_{\text{vol}} \le 300, & \text{(Global Budget)} \\ \displaystyle\sum_{(r,c)\in\mathcal{G}} N_{\text{uav}}(r,c) \le 16, & \text{(UAV Fleet Limit)} \\ N_{\text{vol}} \le 180, \qquad N_{\text{ranger}} \le 300, & \text{(Personal Capacity)} \\ 0 \le N_{\text{cam}}(r,c) \le 3, \quad \forall\,(r,c)\in\mathcal{W}, & \text{(Camera Saturation)} \\ N_{\text{cam}}(r,c) = 0, \qquad \forall\,(r,c)\notin\mathcal{W}, & \text{(Camera Placement)} \\ N_{\text{cam}},\ N_{\text{uav}},\ N_{\text{ranger}},\ N_{\text{vol}} \in \mathbb{Z}_{\ge 0}, & \text{(Integrality)} \end{cases} \tag{5.2}$$

where $\mathcal{W} \subset \mathcal{G}$ denotes the set of waterhole cells within the park grid. Figure 5.1 demonstrates our overarching mathematical framework.

5.5 Model Solution: Cost-Benefit Greedy Optimization

5.5.1 Algorithm Design

The complete resource allocation model is a large-scale, discrete, non-linear optimization problem. The holistic objective function $\mathcal{F}$ evaluates overlapping protection probabilities across thousands of high-

resolution grid cells. Consequently, exact integer programming solvers (e.g., Branch and Bound) are computationally prohibitive due to the exponential state space.

To efficiently navigate this high-dimensional space, we developed a **Cost-Benefit Greedy Algorithm** implemented in C++. The algorithm starts from an empty asset configuration and iteratively selects the single resource deployment action that yields the highest marginal protection gain per unit of budget ($\Delta\mathcal{F}/\text{Cost}$). This approach continues until the budget $R_{\max} = 300$ is exhausted, or asset-specific capacity limits (e.g., maximum 16 UAVs) are reached.

Algorithm 1 Greedy Resource Allocation

Require: Initial solution $\mathcal{S}$, budget B, risk maps P (poaching) and F (fire)
Ensure: Optimized solution $\mathcal{S}^*$
Initialize:
Compute current API and HPI values for all cells
$api \leftarrow \frac{\sum_{i,j} P_{ij} \cdot \text{API}_{ij}}{\sum_{i,j} P_{ij}}$, $hpi \leftarrow \frac{\sum_{i,j} F_{ij} \cdot \text{HPI}_{ij}}{\sum_{i,j} F_{ij}}$
$bii \leftarrow \text{computeBII}(\mathcal{S}_{\text{vol}})$, $F \leftarrow \alpha \cdot api + \beta \cdot hpi + \gamma \cdot bii$
$rem \leftarrow B - \mathcal{S}_{\text{budget}}$
Enforce minimum volunteers:
while $\mathcal{S}_{\text{vol}} < 36$ and $rem \geq C_{\text{vol}}$ **do**
    $\mathcal{S}_{\text{vol}} \leftarrow \mathcal{S}_{\text{vol}} + 6$, $\mathcal{S}_{\text{budget}} \leftarrow \mathcal{S}_{\text{budget}} + C_{\text{vol}}$
    Update bii, F, and rem
end while
Main greedy loop:
while $rem \geq \min(C_{\text{action}})$ **do**
    $best\_gain \leftarrow -\infty$, $best\_action \leftarrow \emptyset$
    Evaluate candidate actions:
    ▷ Volunteers: $\Delta F_{\text{vol}} = \gamma \cdot [\text{BII}(\mathcal{S}_{\text{vol}} + 6) - bii]$
    ▷ Ranger at (r, c): $\Delta F_{\text{rng}} = \alpha \cdot \Delta api(r, c) + \beta \cdot \Delta hpi(r, c)$
    ▷ UAV at (r, c): $\Delta F_{\text{uav}} = \alpha \cdot \Delta api(r, c) + \beta \cdot \Delta hpi(r, c)$
    ▷ Camera at waterhole w: $\Delta F_{\text{cam}} = \alpha \cdot \Delta api(w)$
    if $best\_gain \leq 0$ **then**
        break
    end if
Apply best action:
    Update $\mathcal{S}$ with selected placement
    $rem \leftarrow rem - \text{cost}(best\_action)$
    Recompute api, hpi, bii, and F
    Store step in history
end while
return $\mathcal{S}$

Algorithm 1 outlines the precise execution logic of our C++ simulation. The complete source code is provided in Appendix A.

5.5.2 Optimization Results Analysis

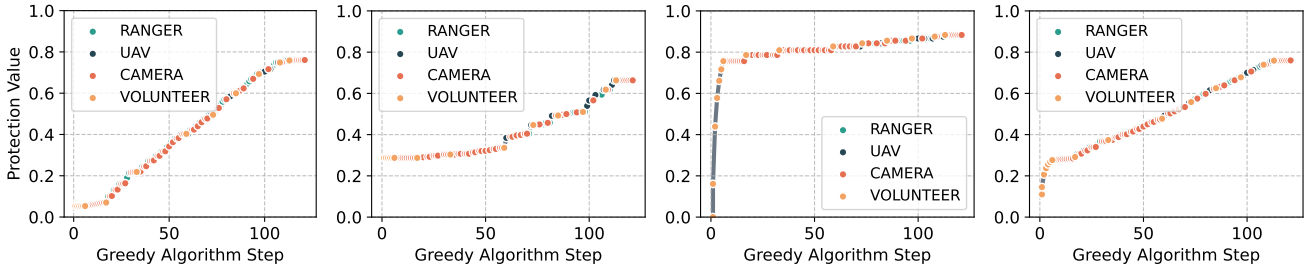
Before executing the greedy optimization, we instantiate the global hyper-parameters derived from Etosha's ecological and logistical data, summarized in Table 5.2. To bridge the gap between abstract code and physical realities, these parameters are systematically categorized into four operational domains: (1) **Objective Weights** (α, β, γ) derived from Problem 1; (2) **Spatio-Temporal Bounds** (e.g., $T^0_{\text{det}}, v_{\text{patrol}}, T_{\text{total}}$) controlling baseline intervention times; (3) **Detection Radii** (e.g., $R_{\text{rng}}, R_{\text{uav}}$) defining

the overlapping 3×3 and 7×7 grid coverage footprints; and (4) **Economic Constraints** defining the unit costs (C) and strict capacity ceilings ($N^{\max}$) for each asset class.

Table 5.2: Etosha National Park: Model Parameters and Baseline Values Setting in Code

Symbol	Value	Symbol	Value	Symbol	Value	Symbol	Value
α	0.488	k_{patrol}	2.0	v_{patrol}	2.0 km/h	C_{ranger}	5.0
β	0.293	P_{sat}	0.25	T_{day}	8.0 h	C_{uav}	7.0
γ	0.220	τ_{fire}	6.0	T_{base}	4.0 h	C_{cam}	0.2
T^0_{det}	6.0 h	$R^{\text{poach}}_{\text{rng}}$	1 (3×3)	T_{total}	720 h	C_{vol}	2.0
κ	1.5	$R^{\text{poach}}_{\text{uav}}$	1 (3×3)	$N^{\max}_{\text{uav}}$	16	N_{grp}	6
σ_{det}	0.8	$R^{\text{fire}}_{\text{rng}}$	1 (3×3)	$N^{\max}_{\text{vol}}$	180	E_{camp}	0.7
w_{center}	1.0	$R^{\text{fire}}_{\text{uav}}$	3 (7×7)	$N^{\max}_{\text{rng}}$	300	$R_{\max}$	300
w_{adj}	0.8	$R^{\text{fire}}_{\text{camp}}$	2 (5×5)	$N^{\max}_{\text{cam,wh}}$	3	$\text{UAV}_{\text{operator}}$	2

*Note: All values are derived or estimated from Etosha-specific data [2].



(a) API (b) HPI (c) BII (d) $\mathcal{F}$ Score

Figure 5.2: Evolution of protection indices during greedy optimization. Each marker color represents a distinct resource type being added. All three sub-indices—API, HPI, and BII—rise monotonically before saturating as budget is exhausted.

Table 5.3: Four Phases of Greedy Optimization (122-Step Specialized Run)

Phase	Steps	$\Delta\mathcal{F}$	Key Characteristics
Perimeter Foundation	1–7	0.166	Initial 6 volunteers (forced) did nothing; steps 1-6 add 36 volunteers, finally activating BII. Volunteers reach threshold where fence patrol becomes effective.
Detection Mesh	8–19	0.017	10 low-cost cameras deployed at 0.2 each, interleaved with 2 volunteer steps. Establishes "tripwire" detection network across high-risk cells.
Ranger-Camera Clusters	20–112	0.490	Each ranger squad placement is followed by 2-3 cameras in adjacent cells. Rangers provide 3×3 patrol strength, cameras boost per-cell detection (60-82%). 39 ranger squads and 9 UAVs deployed in this phase. UAVs inserted when HPI lags behind API.
Budget Exhaustion	113–122	0.004	Final 10 cameras utilize remaining ≈ 4.8 budget with diminishing returns.
Total	**1–122**	**0.677**	**Final protection gain (Final $\mathcal{F}^* = 0.787$ from 0.110 baseline)**

Figure 5.2 and Table 5.3 trace the evolution of the protection indices across the 122 iterations of our algorithmic simulation. The optimization trajectory exhibits four behaviorally distinct phases. Initially, during the **Perimeter Foundation** phase (steps 1–7), the algorithm prioritizes meeting the minimum boundary defense threshold, deploying initial volunteer teams to activate the BII scoring. Subsequently, the **Detection Mesh** phase (steps 8–19) establishes a highly cost-effective "tripwire" network by scattering low-cost AI cameras across high-risk grids.

The core of the optimization occurs in the **Ranger-Camera Clusters** phase (steps 20–112), which generates the most significant protection gain ($\Delta\mathcal{F} = 0.490$). Here, the algorithm dynamically pairs ranger squads (capitalizing on their 3×3 grid interception multiplier) with overlapping cameras, while intermittently injecting UAVs whenever the area-based HPI lags behind point-based API. Finally, as major structural vulnerabilities are secured, the algorithm enters the **Budget Exhaustion** phase (steps 113–122), expending the residual budget on supplementary cameras with diminishing returns.

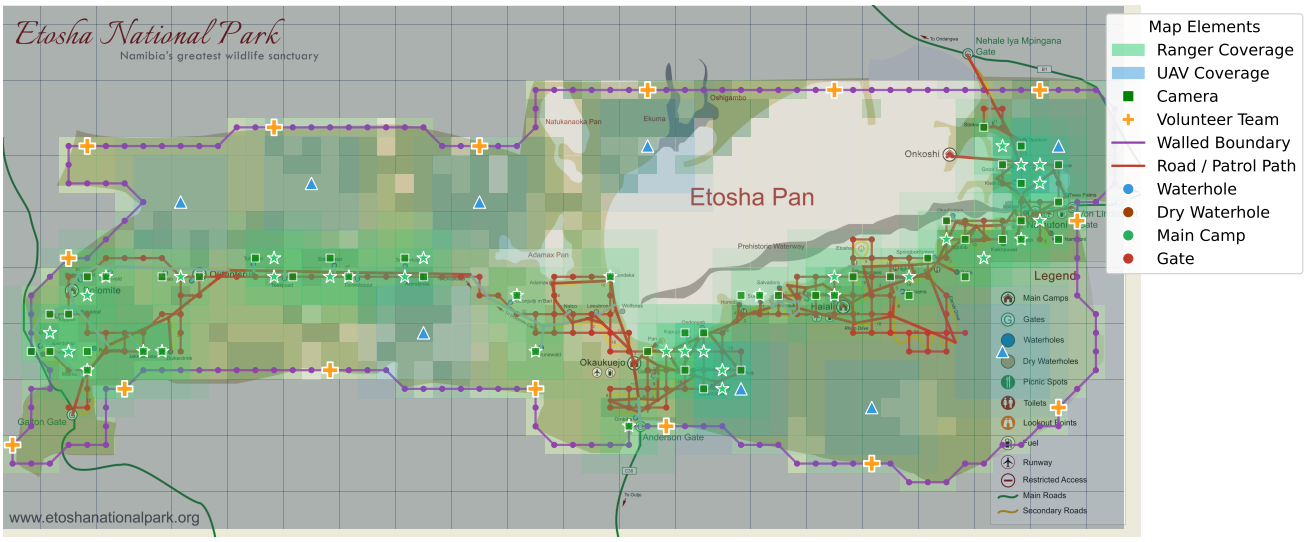


Figure 5.3: **Optimal resource deployment map for Etosha National Park.** The green pentagrams and shaded areas represent 195 rangers, the blue triangles represent 9 UAVs, and the green squares represent 59 cameras around the waterholes. The orange plus signs represent patrol teams comprising 15 volunteers teams responsible for checking the boundaries.

Table 5.4: Optimal Resource Allocation Summary

Resource Type	**Units Deployed**	**Budget Cost**	**$\mathcal{F}$ Contribution**	**Primary Index**
Rangers	195 (39 squads)	195.0	0.345	API
UAVs	9 (54 personnel total)	63.0	0.090	HPI
AI Cameras	59	12.0	0.036	API
Volunteers	90 (15 teams)	30.0	0.194	BII
Baseline	—	0.0	0.122	—
Total	249 personnel	**300.0**	**0.787**	—

Figure 5.3 and Table 5.4 detail the final optimal resource distribution. The deployment achieves **a composite protection score of $\mathcal{F}^* = 0.787$ by completely exhausting the 300-unit budget limit**. The spatial configuration heavily concentrates the **195 rangers (organized into 39 active squads)** and **59 AI cameras** around critical waterhole clusters to maximize poaching interception (API). Meanwhile, the **9 UAVs**, with six personnel per drone (2 per shift, three shifts) are strategically dispersed to provide expansive aerial sweeps over fire-prone savanna regions (HPI), and **90 volunteers (forming 15 cyclic teams)** are distributed along the perimeter to secure the Boundary Integrity Index (BII).

6 Problem 3: Protection Over Time and Staffing Needs

While the model developed in Problem 2 provides a robust static allocation, real-world threat landscapes are highly dynamic. To evaluate protection over time and estimate realistic staffing needs, we introduce two orthogonal temporal dimensions: the **Seasonal Scale** (dry vs. wet) and the **Diurnal Scale** (day vs. night).

During the dry season, wildlife concentrates around 86 waterholes, creating predictable poaching hot spots and elevated wildfire risks. Conversely, the wet season disperses animals and creates mud-clogged roads, severely hindering patrol mobility. On the diurnal scale, nocturnal conditions grant poachers concealment while severely degrading UAV visual detection rates. Combining these dimensions yields four distinct scenarios: $s \in \mathcal{T} = \{\text{Dry-Day, Dry-Night, Wet-Day, Wet-Night}\}$, which are characterized in Figure 6.1.

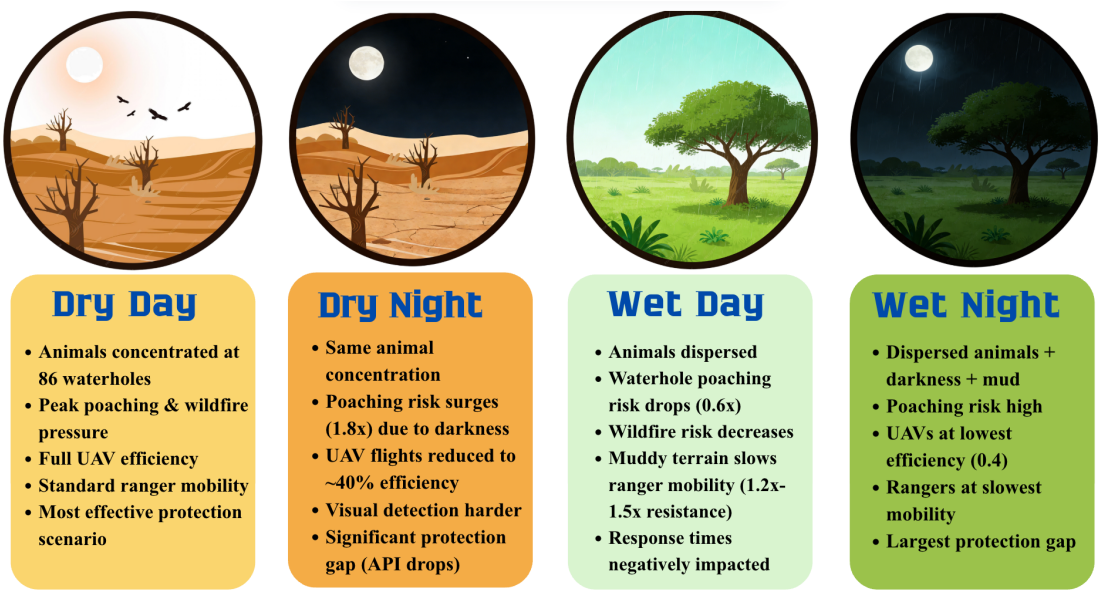


Figure 6.1: Visualization of the four distinct temporal scenarios affecting Etosha's vulnerability.

6.1 Temporal Parameterization and Protection Gaps

We contextualize model by mapping scenario s to shifting exogenous parameters (Table 6.1).

Table 6.1: Dynamic Exogenous Parameters Across Spatio-Temporal Scenarios

Parameter	Physical Meaning	Dry-Day	Dry-Night	Wet-Day	Wet-Night
σ_{det}	Poaching detection probability factor	0.8	0.4	0.8	0.4
R_{detect}	Ranger poaching detection radius (grid cells)	1.0	0.5	1.0	0.5
v_{patrol}	Patrol speed coefficient	2.0	1.0	1.5	0.8
$R_{\text{UAV}}^{\text{fire}}$	UAV fire detection radius (grid cells)	3.0	1.0	3.0	1.0
ϕ_{risk}	Vegetation fire risk multiplier	1.0×	1.0×	0.2×	0.2×
λ_{bio}	Resource/Vegetation abundance multiplier	1.0×	1.0×	1.5×	1.5×

Specifically, nocturnal poaching risk surges ($\eta_{\text{night}} = 1.8$) while UAV sortie effectiveness drops due to visual limitations ($\eta_{\text{uav}} = 0.4$). The wet season disperses targets ($\eta_{\text{wet}} = 0.6$) and increases terrain resistance on standard roads ($\xi_m = 1.5$). By substituting these dynamic parameters into our objective function, we derive the Scenario-Specific Protection Index $\mathcal{F}(s)$.

Forward Evaluation (The Protection Gap). We first test the robustness of the static optimal configuration derived in Problem 2 against these dynamic fluctuations. Let $\mathcal{F}_{\text{static}}(s)$ be the performance

of the static strategy in scenario s. Management dictates a strict baseline protection threshold of $\mathcal{F}^* = 0.700$. The deficit $\text{Gap}(s) = \max(0, \mathcal{F}^* - \mathcal{F}_{\text{static}}(s))$ dictates the need for dynamic reinforcement (see columns 2-3 in Table 6.2). The static allocation fails severely during night-time and wet scenarios, necessitating inverse optimization.

6.2 Inverse Optimization: Minimizing Active Personnel

To meet the $\mathcal{F}^*$ threshold, we invert the logic of Problem 2 shown in Figure 6.2. Removing the fixed manpower constraint, we seek the minimum simultaneous on-duty personnel $H_{\text{active}}(s)$ required for each scenario:

$$\min \quad H_{\text{active}}(s) = \sum_{j \in \mathcal{G}} h_j, \tag{6.1}$$

$$\text{s.t.} \quad \mathcal{F}(s) \geq \mathcal{F}^*, \quad \text{and all equipment/budget constraints defined in Eq. (5.2).}$$

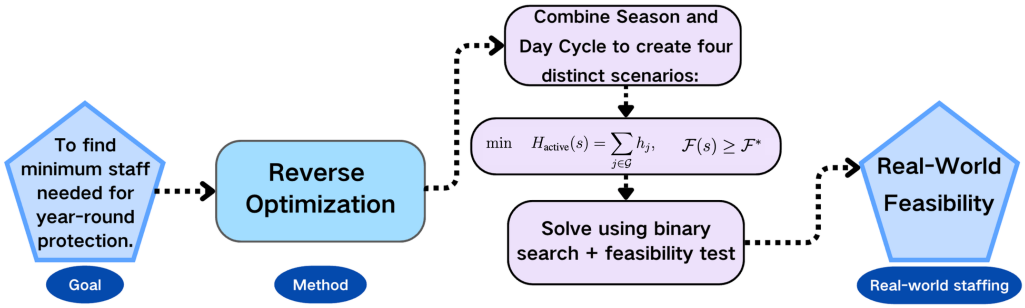


Figure 6.2: Inverse Optimization: Minimizing Active Personnel to meet the $\mathcal{F}^*$ threshold.

Table 6.2: Forward Evaluation Gaps and Inverse Optimization Staffing Requirements

Scenario (s)	$\mathcal{F}_{\text{static}}(s)$	**Gap**	$H^{\min}_{\text{active}}(s)$	**UAVs**	**Total Est. Staff**	**Ratio (Total)**
Dry-Day (Baseline)	**0.787**	0.000	**105**	16	**201**	1.00×
Dry-Night	**0.565**	0.135	**195**	16	**291**	1.45×
Wet-Day	**0.612**	0.088	**125**	16	**221**	1.10×
Wet-Night	**0.560**	0.140	**290**	16	**386**	1.92×

**Note:* $H^{\min}_{\text{active}}(s)$ represents the absolute minimum simultaneous on-duty personnel. The Total Est. Staff is calculated as $H^{\min}_{\text{active}}$ + UAVs × 6 operators, accounting for realistic 24/7 rotational requirements.

We solve this using a **binary search** coupled with our greedy heuristic. The smallest integer H that successfully satisfies the baseline protection threshold forms $H^{\min}_{\text{active}}(s)$. As detailed in Table 6.2, the results highlight the severe operational strain imposed by adverse environmental conditions. Notably, **to maintain the same baseline protection standard during the worst-case Wet-Night scenario, the park requires a workforce multiplier of** $1.92\times$ **compared to the standard Dry-Day allocation.**

7 Problem 4: Sensitivity and Scenario Analysis

The conclusions of Problem 2 rest on a set of calibrated exogenous parameters (κ, μ, α, β, γ, $H_{\max}$, etc.). In practice, these parameters are never known with certainty: AI cameras lose acuity in sandstorms or heavy rain, managers may re-weight ecological priorities after a wildfire season, and budget shocks can suddenly tighten resource ceilings. If the model is highly sensitive to such perturbations, the optimal plan derived in Problem 2 may be fragile in the field.

This section evaluates model robustness from **two complementary angles**:

- **Sensitivity Analysis:** Within a $\pm 20\%$ to $\pm 50\%$ perturbation window, we quantify how the optimal Overall Protection Index $\mathcal{F}^*$ and the underlying budget requirements respond to changes in key exogenous parameters.
- **Scenario Stress Tests:** We impose three extreme real-world constraints-a staffing crunch, a technology supply disruption, and nighttime patrol restrictions-and observe whether the protection system degrades gracefully, validating the strategic triage embedded in the model's objective function.

Sensitivity Metric: The Elasticity Index. To quantify the relative sensitivity of the optimal objective to a parameter p, we introduce the **elasticity index** E_p:

$$E_p = \left| \frac{\Delta \mathcal{F}^* / \mathcal{F}^*}{\Delta p / p} \right|. \tag{7.1}$$

An index $E_p > 1$ (elastic) indicates that a 1% change in p produces more than a 1% change in $\mathcal{F}^*$, meaning p is a **critical vulnerability** requiring precise calibration. Conversely, $E_p < 1$ (inelastic) signals robustness: the model tolerates uncertainty without catastrophic performance swings.

As visually summarized in the **Tornado Charts (Figure 7.1)**, we perturbed major operational and economic parameters. The bounded nature of the charts (with max $\mathcal{F}^*$ fluctuations strictly under 2%) visually confirms the mathematical inelasticity of our model. Even under 50% parameter swings, the optimization algorithm internally compensates by re-routing the abstract budget, ensuring overarching structural stability.

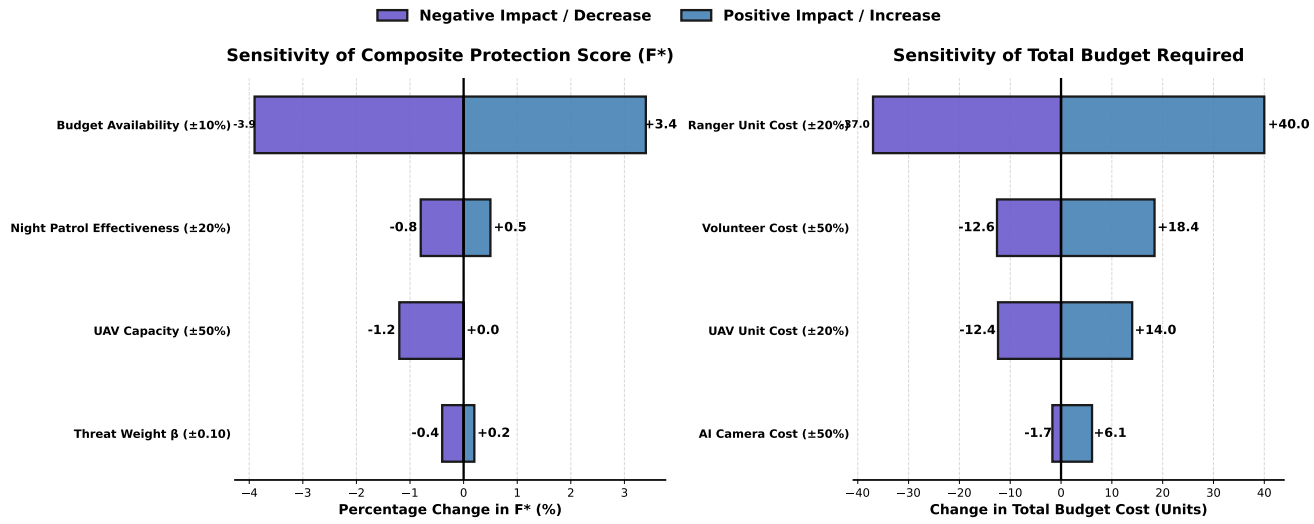


Figure 7.1: Tornado Charts illustrating model sensitivity. The left panel displays the impact of operational perturbations on the composite protection score ($\mathcal{F}^*$). The right panel demonstrates the minimum total budget required to successfully maintain the baseline protection level ($\mathcal{F}^* = 0.787$) under fluctuating resource unit costs.

7.1 Sensitivity Analysis I: Threat Prioritization Weights (α, β, γ)

The baseline weights $(\alpha, \beta, \gamma) = (0.488, 0.293, 0.220)$ derive directly from our Problem 1 risk matrix. To evaluate model robustness against shifting ecological priorities, we perturb the wildfire weight β while adjusting the remaining two proportionally to maintain $\alpha + \beta + \gamma = 1$:

$$\alpha \to \alpha - \frac{\Delta}{2}, \quad \beta \to \beta + \Delta, \quad \gamma \to \gamma - \frac{\Delta}{2},$$

where $\Delta \in [-0.093, +0.207]$.

Analysis: Seamless Budget Migration. As β increases, the model seamlessly shifts resources from

rangers and volunteers to UAVs to prioritize fire detection, as illustrated in Table 7.1. The composite score $\mathcal{F}^*$ remains exceptionally stable (elasticity $E_\beta \approx 0.014$), proving that the greedy allocation strategy adapts organically to top-down policy shifts.

Table 7.1: Resource Allocation Shift under Perturbed Priority Weights

α (Poach)	β (Fire)	γ (Breach)	**Rangers**	**UAVs**	**Cameras**	**Volunteers**	$\mathcal{F}^*$
0.534	0.200	0.266	220	5	55	102	0.796
0.488	0.293	0.220	195	9	59	90	**0.787**
0.434	0.400	0.166	170	13	50	73	0.783
0.385	0.500	0.115	165	15	50	60	0.792

7.2 Sensitivity Analysis II: Scenario Stress Tests

Scenario A: Severe Personnel Reduction. We simulate a 50% reduction in maximum ranger availability ($H_{\max} = 300 \to 150$). As the primary mobile response force is restricted, the optimization algorithm undergoes a forced strategic triage, reallocating limited human resources away from low-priority zones to safeguard critical assets.

Table 7.2: Stress-test analysis under ranger and UAV reductions

(a) Impact of 50% Ranger Reduction on Protection Metrics

Metric	**Baseline**	**Scenario A**	**Change**
$\mathcal{F}^*$	0.787	0.772	-1.91%
API	0.764	0.669	-12.43%
HPI	0.749	0.840	$+12.15\%$
BII	0.883	0.906	$+2.60\%$

(b) Impact of 75% UAV Reduction on Protection Metrics

Metric	**Baseline**	**Scenario B**	**Change**
$\mathcal{F}^*$	0.787	0.773	-1.78%
API	0.764	0.804	$+5.24\%$
HPI	0.749	0.625	-16.56%
BII	0.883	0.896	$+1.47\%$

We define the **Degradation Ratio** to measure systemic resilience against labor shortages. The disproportionate drop in the Anti-Poaching Index (API) highlights the model's reliance on human intervention for point-based protection:

$$\text{Degradation Ratio} = \frac{|\Delta\text{API}|/\text{API}}{|\Delta\mathcal{F}^*|/\mathcal{F}^*} = \frac{12.43\%}{1.91\%} \approx 6.51$$

Interestingly, both HPI (+12.15%) and BII (+2.60%) increase. This indicates that the algorithm intelligently compensates for mobile personnel scarcity by reallocating the abstract budget into aerial surveillance (UAVs) and static boundary defense, where human presence yields higher marginal returns per capita.

Scenario B: Technological Asset Deficit. Quartering the UAV capacity ($Y_{\max} = 16 \to 4$) fundamentally cripples aerial surveillance capabilities. This predominantly impacts rapid fire detection across the expansive savanna grid. We quantify this vulnerability using the **Technological Dependence Index (TDI)**. For area-based wildfire monitoring:

$$\text{TDI}_{\text{fire}} = \frac{|\Delta\text{HPI}|/\text{HPI}}{|\Delta Y_{\max}|/Y_{\max}} = \frac{16.56\%}{75.00\%} \approx 0.22$$

A TDI of 0.22 implies that every 10% reduction in UAV availability translates to a 2.2% direct loss in habitat protection, underscoring the indispensability of aerial technology for area-based threats.

Scenario C: Nighttime Patrol Restrictions. Nocturnal operations face severely reduced patrol effectiveness and elevated occupational risks. To maintain a strict nighttime holistic performance threshold of $\mathcal{F}^*_{\text{night}} \geq 0.90 \times \mathcal{F}^*_{\text{day}}$, the optimization model dynamically compensates. By substituting

human mobility with static surveillance, the model heavily funnels the budget into AI-assisted cameras to bridge the detection gap from 59 to 74. This demonstrates that under environmental stress, capital-intensive technology behaves as a highly elastic substitute for labor mobility.

8 Problem 5: Model Strengths and Limitations

8.1 Strengths

1. **Unified Grid-Based Spatial Triad (Point-Area-Line):** The model transforms the abstract concept of "protection" into a quantifiable, grid-computed framework. Discretizing the 22,935 km^2 park into uniform cells, it preserves the distinct geographical nature of each threat type. The Anti-Poaching Index (API) operates as a *point-focused* metric, where protection radiates outward from discrete assets deployed at specific coordinates. The Habitat Protection Index (HPI) serves as a true *area-based* monitor, aggregating detection probabilities across continuous vulnerable biomes. The Boundary Integrity Index (BII) secures the *line-based* perimeter through segmented integrity calculations. This architecture allows heterogeneous threats to be evaluated within a single overarching matrix.
2. **Probabilistic Redundancy and Endogenous Prioritization:** Rather than relying on deterministic coverage assumptions, the model evaluates protection through a **joint probability framework** (e.g., $P_{\text{detect}} = 1 - \prod(1 - p_i)$), realistically capturing the synergistic effects of overlapping sensor networks and correctly modeling the diminishing marginal returns of asset stacking. By embedding geographic modifiers directly into the grid's risk weights ($R_{\text{poach}}, R_{\text{fire}}$), the model achieves endogenous spatial prioritization—naturally gravitating toward critical vulnerabilities without manually hard-coded exclusion zones.
3. **Computational Tractability and Operational Translation:** Solving a combinatorial resource allocation problem over thousands of grid cells is typically prohibitive. Our Cost-Benefit Greedy Algorithm navigates this MINLP space efficiently by evaluating the marginal gain-to-cost ratio ($\Delta\mathcal{F}/\text{Cost}$) at each step. Critically, all constraints are grounded in operational reality—accounting for shift multipliers, distinct asset costs, and finite deployment capacities—ensuring the mathematical output is not merely a theoretical optimum but an integer-based deployment blueprint directly actionable by park management.

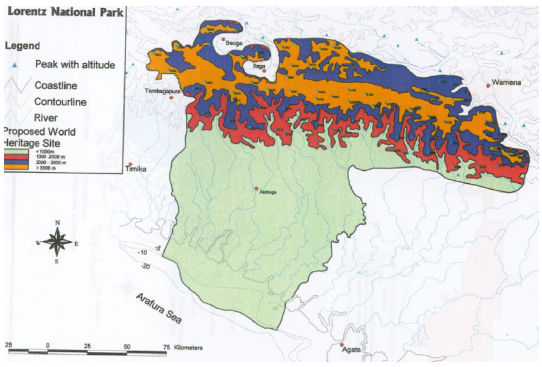
8.2 Weaknesses and Limitations

1. **High Dependency on Empirical Calibration Data:** The model's sophistication requires several exogenous parameters that are difficult to estimate with certainty. The base detection conversion rate (k_{patrol}), sensor coverage radii (e.g., the 3×3 grid assumption for rangers versus 7×7 for UAVs), and biome resistance values all materially influence the final deployment shape. While our sensitivity analysis (Problem 4) demonstrated systemic resilience at the aggregate level, localized grid-level deployment decisions remain sensitive to these calibration assumptions.
2. **Static Threat Mapping vs. Adaptive Adversaries:** Risk topographies are computed from static geographic attractors such as proximity to waterholes and roads. Poaching, however, is fundamentally adversarial. If management heavily fortifies waterholes, rational poachers will shift routes and target less-guarded zones. The model optimizes against a historical "snapshot" of risk, lacking the dynamic game-theoretic mechanisms needed to simulate an adversary that learns and reacts to observed patrol patterns over time.
3. **Myopic Nature of the Greedy Heuristic:** While the greedy algorithm is computationally efficient and provides strong approximation guarantees for submodular functions, it is fundamentally myopic. Evaluating only the immediate best ratio ($\Delta\mathcal{F}/\text{Cost}$) at each step risks settling into local optima. Our implementation partially mitigates this through emergent synergistic behavior—the model consistently exhibits a "ranger-first, cameras-follow" pattern that captures complementary asset interactions. Nevertheless, the step-by-step approach may still miss complex multi-asset strategies requiring simul-

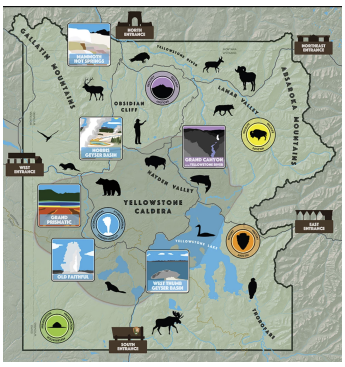
taneous deployment to unlock non-linear protection gains.

9 Problem 6: Model Adaptation to Other Protected Areas

To demonstrate the universality and transferability of our grid-based spatial triad model, we adapt our framework to two distinctly different environments: **Lorentz National Park** (a tropical, mountainous rainforest in Indonesia) and **Yellowstone National Park** (a temperate alpine ecosystem in the USA).



(a) Lorentz National Park Map



(b) Yellowstone National Park Map

Figure 9.1: The Comparison Between the Two National Parks Located on Different Continents.

9.1 Universal Model Components (What Remains Unchanged)

The architectural core of our operations research model is geographically agnostic and remains strictly unchanged:

- **Grid-Based Spatial Triad ($\mathcal{G}$):** The discretization of the park into computational grid cells and the division of threats into Point (API), Area (HPI), and Line (BII) indices.
- **Probabilistic Redundancy Logic:** The joint detection probability framework ($P_{\text{detect}} = 1 - \prod(1 - p_i)$), which mathematically handles overlapping sensor networks.
- **Cost-Benefit Greedy Optimization:** The algorithmic solver evaluating the marginal gain-to-cost ratio ($\Delta\mathcal{F}/\text{Cost}$) to distribute constrained resources remains universally applicable.

9.2 Adaptation 1: Lorentz National Park, Indonesia (Tropical / Montane)

Lorentz National Park spans 25,056 km^2 across an extreme elevation gradient. Conservation challenges shift from waterhole-centric poaching to forest poaching of deer, wild pigs, and cassowaries, alongside illegal logging and mining encroachment.

Modified Parameters & Inputs:

- **Terrain Impedance:** Dramatic elevation changes and dense vegetation require resistance matrix recalibration using digital elevation models, with steeper slopes receiving higher impedance penalties and swampy lowlands and montane forests become effectively impassable for ground patrols.
- **Detection Technology Substitution:** Persistent cloud cover and multi-layered canopy render optical cameras largely ineffective. We replace them with **acoustic AI sensors** (same cost and 1×1 coverage) calibrated to detect chainsaw frequencies and human voices.
- **Point-Target Redefinition (ρ_{poach}):** Risk weights shift to river confluences, logging camp access points, and high-elevation monsoon refuges, replacing waterhole-centric API calculations.

Calibration Data Required: High-resolution DEMs for slope-based impedance, satellite LiDAR

canopy density maps, and ranger incident records to redefine risk weights.

Expected Deployment Shift: Optical camera investment decreases; acoustic sensors concentrate at strategic chokepoints. UAVs shift from broad visual surveillance to targeted nocturnal thermal scanning of river corridors. Ranger allocation focuses on lowland areas, with reduced high-elevation presence.

9.3 Adaptation 2: Yellowstone National Park, USA (Temperate / Alpine)

Yellowstone's 8,983 km^2 presents a fundamentally different landscape shaped by extreme seasonal variation, over 4 million annual visitors, and wildfire risk. Poaching pressure is minimal; human-wildlife conflict and fire management become the primary concerns.

Modified Parameters & Inputs:

- **Objective Weight Recalibration:** Weights shift from $(\alpha, \beta, \gamma) = (0.488, 0.293, 0.220)$ to $(\alpha', \beta', \gamma') = (0.25, 0.55, 0.20)$, significantly increasing fire's importance while reducing anti-poaching emphasis.
- **Seasonal Wildlife Aggregation:** Winter aggregation in geothermal basins replaces Etosha's dry-season waterhole concentration. GPS collar data from bison and elk reveals consistent low-elevation wintering grounds where snow does not accumulate, concentrating ρ_{poach} weights seasonally in these thermal zones.
- **Human-Wildlife Conflict Integration:** Roads and high-visitor areas receive elevated risk weights reflecting bison-vehicle collision and bear-human conflict potential, informed by visitor density maps.
- **Fire Risk Dynamics (ϕ_{fire}):** Wildfire risk depends on timber moisture, beetle-kill accumulation, and lightning strike density rather than savanna grass dryness. Risk weights are recalibrated using forest mortality maps and historical ignition patterns, with β spiking sharply during late-summer drought.

Calibration Data Required: Ungulate GPS telemetry for seasonal aggregation mapping, bark beetle forest mortality surveys, historical lightning strike databases, and visitor traffic heatmaps.

Expected Deployment Shift: The model exhibits pronounced seasonal phase-shifting. Summer prioritizes UAV swarms for wildfire detection and ranger presence along high-traffic corridors. Winter shifts resources to thermal basin wildlife monitoring, with fire detection investment reduced as snow eliminates ignition risk. Camera placement concentrates on roads rather than backcountry.

Table 9.1: Comparative Parameter Mapping Across Distinct Global Ecosystems

Modeling Component	Etosha (Arid Savanna)	Lorentz (Tropical Montane)	Yellowstone (Temperate Alpine)
Point Targets (API)	Waterholes, road intersections	River confluences, logging camps	Thermal basins (winter), roads (year-round)
Terrain Matrix	Moderate (Mud/Pans)	High (Elevation gradient)	Variable (Deep snow, slopes)
Detection Technology	Optical cameras	Acoustic sensors (chainsaw/voice)	Optical cameras (roads/trails)
Aerial Efficacy (η_{uav})	High (Open sky)	Moderate (Thermal through canopy)	Weather-dependent
Primary Area Risk	Dry savanna grasses	Illegal deforestation	Timber fires, beetle kill
Seasonal Drivers	Dry vs. Wet Season	Monsoon flooding	Winter aggregation vs. Summer fire
Additional Factors	—	Elevation limits patrol access	Visitor density affects conflict risk

10 Communication Deliverable: Letter to the IMMC

Dear Members of the IMMC,

We deeply appreciate the opportunity to present our findings regarding the conservation of Etosha National Park. Managing a sanctuary of nearly 23,000 square kilometers with finite resources is a monumental task. Park administrators are constantly forced to make difficult trade-offs between protecting endangered wildlife, preserving vast habitats, and maintaining essential infrastructure. Recognizing the weight of these responsibilities, our team set out to design a comprehensive, data-driven framework to support your conservation efforts.

Our analysis suggests that attempting to distribute personnel uniformly across such a massive landscape often dilutes overall effectiveness. Instead, we respectfully propose a **"Spatial Triad" protection system** as illustrated in Figure 10.1. By mathematically dividing the park into a high-resolution grid, this framework allows park managers to transition from blanket patrols to precise, strategic triage.

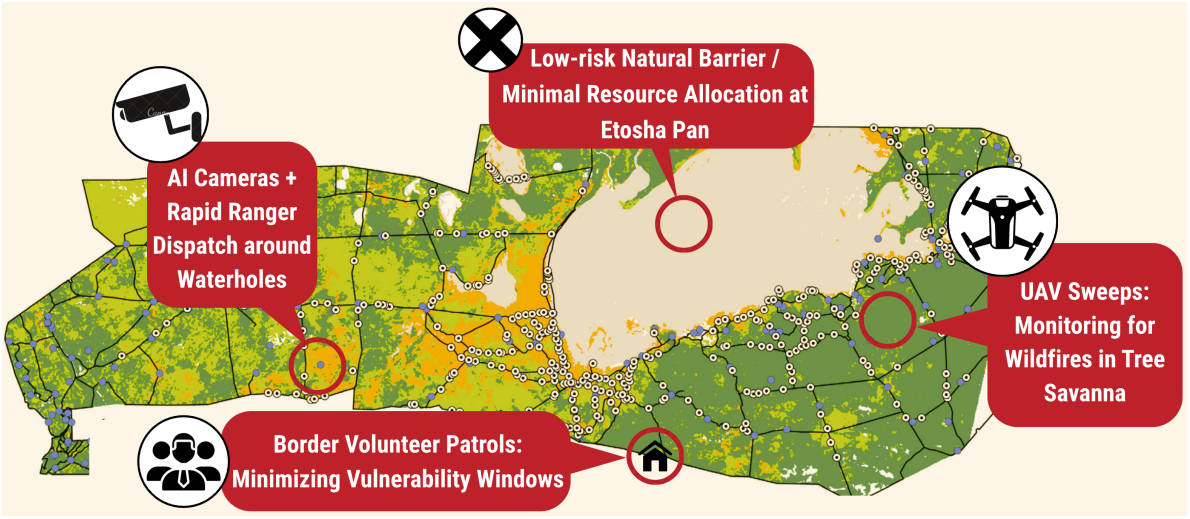


Figure 10.1: Strategic spatial optimization: The Point-Area-Line deployment dashboard.

Our Approach: The Spatial Triad

We have translated our spatial modeling into an intuitive operational logic, categorizing the park's complex threats into three distinct geographical dimensions:

1. **Points (Poaching): Layered Detection and Interception.** While wildlife and poachers move across the entire landscape, the fundamental logic of anti-poaching remains point-anchored-resources deployed at strategic locations create protective influence that radiates outward. Our model deploys a layered security web where AI-powered cameras provide relentless, passive detection, acting as an early-warning tripwire. This ensures that when highly trained ranger squads are deployed, they are actively intercepting confirmed threats rather than searching blindly.
2. **Areas (Wildfires): Overlapping Aerial Surveillance.** The vast savannas require continuous monitoring to prevent catastrophic wildfires. Because ground patrols often suffer from obstructed views due to dense brush, we utilize Unmanned Aerial Vehicles (UAVs) to cast a wide, overlapping surveillance net. This aerial perspective minimizes the probability that a newly ignited fire goes unnoticed.
3. **Lines (Boundaries): Cyclic Volunteer Patrols.** The 850 km perimeter fence is the park's primary filter against human-wildlife conflict. Instead of constructing expensive permanent outposts, our strategy suggests deploying organized teams of volunteers on continuous, cyclic patrols. This active human presence efficiently shrinks the time window during which a broken fence remains vulnerable.

Profound Insights from the Optimization Engine

Beyond standard resource counting, our algorithmic simulations have yielded three deeper insights that we believe are crucial for shaping long-term conservation strategies:

- **The Synergy of Heterogeneous Assets.** No single resource can protect the park. Our findings indicate that the highest protection scores are not achieved by simply buying more cameras or hiring more rangers, but by ensuring they overlap. For instance, pairing a fixed AI camera (for instant detection) with a nearby mobile ranger squad (for rapid interception) yields a non-linear leap in security effectiveness.
- **Technology as a Labor Substitute Under Environmental Stress.** While skilled rangers are the irreplaceable core of conservation, their mobility plummets during extreme environmental conditions, such as nighttime hours or wet-season floods. Our stress tests reveal that capital-intensive technology behaves as a highly elastic substitute for labor mobility. Investing in all-weather, night-capable UAVs and cameras acts as an essential operational bridge when human patrols face physical limitations.
- **The Mathematical Virtue of "Strategic Neglect."** A profound takeaway from our optimization model is the validation of strategic triage. The algorithm organically deprioritizes vast, inhospitable regions (like the deep Etosha Salt Pan) because these natural barriers inherently deter both wildlife and poachers. By having the confidence to accept zero coverage in these naturally buffered zones, management can free up critical budget to heavily fortify the most vulnerable access points.

Actionable Recommendations

Based on the outcomes of our simulations, we humbly offer the following recommendations for your consideration:

1. **Adopt the Optimal Baseline Configuration:** Subject to your 300-unit budget constraint, our model suggests allocating resources toward **39 active ranger squads (195 personnel)**, **9 long-range UAVs (54 personnel)**, **59 AI cameras** clustered at high-risk waterholes, and **15 volunteer teams (90 individuals)** for boundary maintenance. We believe this specific combination maximizes the return on your operational investment.
2. **Embrace Seasonal Resource Agility:** Rather than maintaining a rigid, year-round operational posture, our temporal analysis indicates that risks shift dramatically with the seasons. We recommend exploring flexible deployments-such as intensifying aerial surveillance during the dry season's peak wildfire window, or augmenting ranger presence near roads when wet-season floods restrict interior access.
3. **Formalize the Ranger-Camera Synergy as Standard Protocol:** Our optimization consistently revealed a powerful pattern: the highest marginal gains occurred when camera deployments immediately followed ranger placements in adjacent cells. We recommend formalizing this protocol - whenever a new ranger squad is deployed to a sector, allocate budget for accompanying AI cameras in surrounding high-risk cells. This exploits the mathematical synergy where rangers provide wide-area patrol strength while cameras deliver precise per-cell detection, yielding significantly more protection than deploying either asset in isolation.

We hope this spatial triad framework offers a flexible, insightful, and practical template not only for Etosha but for other large-scale conservation efforts globally. Thank you for your time, consideration, and unwavering dedication to preserving our natural heritage.

Respectfully submitted,

Team #IMMC2026033

References

[1] Etosha National Park Wall Map. https://www.honeyguidepublications.com/wall-maps/etosha-national-park-wall-map.

[2] Kilian W, Kolberg H. Aerial survey of Etosha National Park[J]. Internal Rep. Min. Environ. Tour., 2015.

[3] Le Roux C J G, Grunow J O, Bredenkamp G J, et al. A classification of the vegetation of the Etosha National Park[J]. South African Journal of Botany, 1988, 54(1): 1-10.

[4] Ferreira S M, Bissett C, Cowell C R, et al. The status of rhinoceroses in South African national parks[J]. Koedoe, 2017, 59(1): 1-11.

[5] Yellowstone National Park Map. https://www.nps.gov/yell/planyourvisit/maps.htm

[6] Greater Yellowstone Ecosystem. https://greateryellowstone.org/greater-yellowstone-map?gad_source=1&gad_campaignid=1348965038&gbraid=0AAAADtyZvXO6P_6Vj cgmG07z7fLfYNNg&gclid=Cj0KCQiAk6rNBhCxARIsAN5mQLvBWyVHbvXZbKoIN j3HpSxX-dCmrPiLxtG2JxY8endorl9brNfgyPgaAo2pEALw_wcB

[7] Lorentz National Park. https://worldheritageoutlook.iucn.org/explore-sites/lorentz- national-park

[8] Lorentz National Park. https://www.worldheritagesite.org/list/lorentz-national-park/

[9] Lorentz National Park. https://whc.unesco.org/en/list/955/maps/

[10] Mammals. https://www.nps.gov/yell/learn/nature/mammals.htm

[11] Park Facts. https://www.nps.gov/yell/planyourvisit/parkfacts.htm

[12] Etosha National Park Map. https://www.researchgate.net/figure/Map-of-the-study-area-Etosha-National-Park-Namibia-The-locations-of-sampling-sitesfig1318895272

[13] Security+Anti-Poaching. https://www.lewa.org/security-read-more/

[14] Connected Conservation. https://thedocs.worldbank.org/en/doc/72c3598b88de14ead53dc18a82 e6c0e7-0320052022/original/poaching-connected-conservation.pdf'

[15] WIldland Fire Smoke in the United States https://www.fs.usda.gov/research/publications/book/ wildfiresmoke/wildfiresmokefull.pdf

[16] News and Event. https://www.iucn.org/news/protected-areas/202407/new-dawn-protected-area-management-uav-technology-central

[17] Evaluating the Use of UAVs for Anti-Poaching. https://www.researchgate.net/publication/337777267

[18] Chu, T.; Starek, M.J.; Berryhill, J.; Quiroga, C.; Pashaei, M. Simulation and Characterization of Wind Impacts on sUAS Flight Performance for Crash Scene Reconstruction. UAVs 2021, 5, 67.

[19] Kass, J. M., Muscarella, R., Galante, P. J., Bohl, C. L., Pinilla-Buitrago, G. E., Boria, R. A., Soley-Guardia, M., & Anderson, R. P. (2021). ENMeval 2.0: Redesigned for customizable and reproducible modeling of species' niches and distributions. Methods in Ecology and Evolution, 12(9), 1602-1608.

[20] Fangjun Li, Xiaoyang Zhang, Shobha Kondragunta, Christopher C. Schmidt, Christopher D. Holmes, A preliminary evaluation of GOES-16 active fire product using Landsat-8 and VIIRS active fire data, and ground-based prescribed fire records, Remote Sensing of Environment, Volume 237, 2020, 111600, ISSN 0034-4257.

[21] Poaching. Hotspotshttps://www.researchgate.net/publication/320009347

[22] Fence Management. https://karingani.com/conservation/fence-management/

Appendices

AI Use Report

1. Tools and Models Utilized

Our team utilized Deepseek-R1 as the primary large language model (LLM) for assistance throughout the project.

2. Primary Functions and Scope of Use

Deepseek AI was employed for two main functions:

1. **Grammatical Refinement and Text Streamlining:** The tool was used to review and edit written content, including report drafts and mathematical explanations. Its primary task was to improve clarity, correct grammatical errors, and ensure overall coherence without altering the original meaning or technical accuracy.
2. **Code Autocompletion and Debugging:** The model assisted in generating code snippets, offering context-aware autocompletion suggestions, and helping to identify and resolve logical errors within our team's custom scripts. This was particularly valuable for debugging our visualization code.

3. Implementation and Access

The tool was accessed via its official web interface. All interactions were conducted in English.

4. Statement on Original Work

Our team exercised complete oversight throughout the process. All AI-generated content, whether textual or code, was rigorously reviewed and validated by our team. While Deepseek-R1 served as an assistive tool, the final project outcomes, conclusions, and intellectual contributions remain the original work of this team.

Appendix A Code

C++: Greedy Resource Allocation Algorithm

```
#include <vector>
#include "Greedy.h"
#include "Params.h"
#include "DataLoader.h"
#include "Calculations.h"
using namespace std;

Solution initSolution(){
    Solution s;
    s.rangers_grid.assign(rows, vector<int>(cols, 0));
    s.cameras.assign(nodes.size(), 0);
    s.uav_grid.assign(rows, vector<int>(cols, 0));
    s.volunteers = 0;
```

```
    // Trackers
    s.budget_used = 0;
    s.total_rangers = 0;
    s.total_cameras = 0;
    s.total_uav = 0;

    // Compute initial F value (baseline with no resources)
    s.current_f = computeF(s);
    s.f_history.push_back(s.current_f);

    return s;
}
```

```
#include "Calculations.h"
#include "DataLoader.h"
#include "Greedy.h"
#include "Params.h"
#include <vector>
#include <cmath>
using namespace std;

float getPatrolStrength(int r, int c,
                        const vector<vector<int>>& rangers_grid) {
    float strength = 0;

    // Rangers in 3x3
    for (int dr = -1; dr <= 1; dr++) {
        for (int dc = -1; dc <= 1; dc++) {
            int nr = r + dr;
            int nc = c + dc;
            if (nr >= 0 && nr < rows && nc >= 0 && nc < cols) {
                float w = weights.pw_3[dr+1][dc+1];
                strength += rangers_grid[nr][nc] * w;
            }
        }
    }

    // Add precomputed camp patrol coverage
    strength += camp_patrol_coverage[r][c];

    return strength;
}

float getPoacherDetection(int r, int c,
                          const vector<vector<int>>& rangers_grid,
                          const vector<vector<int>>& uav_grid,
                          const vector<int>& cameras) {

    // 1. Patrol detection from rangers + camps
    float patrol_strength = getPatrolStrength(r, c, rangers_grid);
    float patrol_detect = 1 - exp(-p_params.PATROL_K * patrol_strength);

    // 2. UAV detection (5x5 for poachers)
    float uav_strength = 0;
    int rad = p_params.UAV_D_RADIUS;
    for (int dr = -rad; dr <= rad; dr++) {
        for (int dc = -rad; dc <= rad; dc++) {
            int nr = r + dr, nc = c + dc;
            if (nr >= 0 && nr < rows && nc >= 0 && nc < cols) {
                float w = weights.pw_5[dr+rad][dc+rad];  // 5x5 weights for UAVs
                uav_strength += uav_grid[nr][nc] * w;
            }
```

```
        }
    }
    float uav_detect = 1 - exp(-p_params.PATROL_K * uav_strength);

    // 3. Camera detection
    float camera_detect = 0;
    for (int wh_id : waterhole_ids) {
        auto it = waterhole_grid.find(wh_id);
        if (it != waterhole_grid.end() && it->second.first == r && it->second.second == c)
            int cams = cameras[wh_id];
            float mu_det = p_params.T_DET_0 / (1 + p_params.KAPPA * cams);
            camera_detect = max(camera_detect, 1 - mu_det / p_params.T_DET_0);
        }
    }

    // 4. Combined probability
    return 1 - (1 - patrol_detect) * (1 - uav_detect) * (1 - camera_detect);
}

float getFireDetection(int r, int c,
                       const vector<vector<int>>& rangers_grid,
                       const vector<vector<int>>& uav_grid) {

    // Start with satellite baseline miss probability
    float miss_prob = 1 - f_params.SAT_BASE;
    // 1. Rangers
    int rad_r = f_params.RANGER_F_RADIUS;
    for (int dr = -rad_r; dr <= rad_r; dr++) {
        for (int dc = -rad_r; dc <= rad_r; dc++) {
            int nr = r + dr, nc = c + dc;
            if (nr >= 0 && nr < rows && nc >= 0 && nc < cols && rangers_grid[nr][nc] > 0)
            {
                float w = weights.fw_5[dr+rad_r][dc+rad_r];
                float detect = 1 - exp(-f_params.PATROL_K * rangers_grid[nr][nc] * w);
                miss_prob *= (1 - detect);
            }
        }
    }
    // 2. UAVs
    int rad_u = f_params.UAV_F_RADIUS;
    for (int dr = -rad_u; dr <= rad_u; dr++) {
        for (int dc = -rad_u; dc <= rad_u; dc++) {
            int nr = r + dr, nc = c + dc;
            if (nr >= 0 && nr < rows && nc >= 0 && nc < cols && uav_grid[nr][nc] > 0)
            {
                float w = weights.fw_7[dr+rad_u][dc+rad_u];
                float detect = 1 - exp(-f_params.PATROL_K * uav_grid[nr][nc] * w);
                miss_prob *= (1 - detect);
            }
        }
    }
    // 3. Camps (precomputed)
    miss_prob *= (1 - camp_fire_coverage[r][c]);
    // 4. Lookouts (precomputed)
    miss_prob *= (1 - lookout_coverage[r][c]);
    return 1 - miss_prob;
}

float getCellAPIValue(int r, int c,
                      const vector<vector<int>>& rangers_grid,
                      const vector<vector<int>>& uav_grid,
                      const vector<int>& cameras) {
```

```
    float risk = cell_poach_risk[r][c];
    if (risk == 0) return 0;

    float patrol_strength = getPatrolStrength(r, c, rangers_grid);
    float detect = getPoacherDetection(r, c, rangers_grid, uav_grid, cameras);

    return risk * detect * patrol_strength;
}

float getCellHPIValue(int r, int c,
                      const vector<vector<int>>& rangers_grid,
                      const vector<vector<int>>& uav_grid) {

    float risk = cell_fire_risk[r][c];
    if (risk == 0) return 0;

    float detect = getFireDetection(r, c, rangers_grid, uav_grid);

    return risk * detect;
}

float computeBII(int volunteers) {
    if (volunteers == 0) return 0;

    // Each volunteer group covers: speed * time per day
    float groups = volunteers / b_params.PATROL_GROUP_SIZE;
    float daily_coverage_km = groups * b_params.PATROL_SPEED * b_params.PATROL_TIME;

    // How many days to cover entire fence
    float cycle_days = total_fence_km / daily_coverage_km;

    // Hours until a breach is found + repaired
    float T_repair = cycle_days * 24 + b_params.BASE_TIME;

    // For each fence segment
    float bii = 0;
    for (auto& seg : fences) {
        float i_k = max(0.0f, 1 - (seg.lambda_k * T_repair / b_params.TOTAL_TIME));
        bii += seg.w_k * i_k;
    }

    return bii;
}

float computeAPI(const Solution& sol) {
    float total = 0;
    float weight_sum = 0;

    for (int r = 0; r < rows; r++) {
        for (int c = 0; c < cols; c++) {
            float risk = cell_poach_risk[r][c];
            if (risk > 0) {
                float val = getCellAPIValue(r, c, sol.rangers_grid, sol.uav_grid,
                sol.cameras);
                total += risk * val;
                weight_sum += risk;
            }
        }
    }

    return (weight_sum > 0) ? total / weight_sum : 0;
```

```
}

float computeHPI(const Solution& sol) {
    float total = 0;
    float weight_sum = 0;
    float max_risk = 0;

    for (int r = 0; r < rows; r++) {
        for (int c = 0; c < cols; c++) {
            float risk = cell_fire_risk[r][c];
            if (risk > 0) {
                float val = getCellHPIValue(r, c, sol.rangers_grid, sol.uav_grid);
                total += risk * val;
                weight_sum += risk;
                if (risk > max_risk) max_risk = risk;
            }
        }
    }

    cout << "Max fire risk: " << max_risk << ", weight_sum: " << weight_sum << endl;
    return (weight_sum > 0) ? total / weight_sum : 0;
}

float computeF(const Solution& sol) {
    float api = computeAPI(sol);
    float hpi = computeHPI(sol);
    float bii = computeBII(sol.volunteers);

    return weights.ALPHA * api + weights.BETA * hpi + weights.GAMMA * bii;
}
```

```
#include "DataLoader.h"
#include <fstream>
#include <iostream>
#include <vector>
#include <map>
#include <string>
#include "json.hpp"
#include <queue>
#include <functional>
using json = nlohmann::json;
using namespace std;

// Define global variables
vector<Node> nodes;
vector<int> waterhole_ids;
vector<int> gate_ids;
vector<int> camp_ids;
vector<int> lookout_ids;
vector<Edge> edges;
map<pair<int, int>, GridCell> grid;
map<int,pair<int,int>> waterhole_grid;
vector<vector<float>> cell_poach_risk;
vector<vector<float>> cell_fire_risk;
vector<pair<int,int>> camp_cells;
vector<vector<float>> camp_patrol_coverage;
vector<vector<float>> camp_fire_coverage;
vector<pair<int,int>> lookout_cells;
vector<vector<float>> lookout_coverage;
map<int,pair<int,int>> dry_waterhole_grid;
vector<int> dry_waterhole_ids;
float total_fence_km;
```

```
double px_per_km;
double cf;
int rows, cols;
vector<FenceSegment> fences;
void loadAllData() {
    ifstream f1("network_export.json");
    json network_data = json::parse(f1);

    ifstream f2("final_grid_data.json");
    json grid_data = json::parse(f2);

    ifstream f3("calibration.json");
    json calib = json::parse(f3);

    px_per_km = calib["px_per_km"];
    double img_width = calib["original_base_dims"][0].get<double>() / px_per_km;
    double rough_width = 350.0;
    cf = img_width / rough_width;

    // Get grid dimensions
    rows = grid_data["metadata"]["rows"];
    cols = grid_data["metadata"]["cols"];

    // Load nodes
    for (auto& [key, val] : network_data["nodes"].items()) {
        int id = stoi(key);
        Node node;
        node.id = id;
        node.name = val.value("name", "");
        node.type = val["type"];

        auto km = val["km"];
        double rough_x = km[0];
        double rough_y = km[1];

        node.rough_x = rough_x;
        node.rough_y = rough_y;
        node.x = rough_x * cf;
        node.y = rough_y * cf;

        nodes.push_back(node);

        if (node.type == "WH") waterhole_ids.push_back(id);
        else if (node.type == "GA") gate_ids.push_back(id);
        else if (node.type == "MC") camp_ids.push_back(id);
        else if (node.type == "LO") lookout_ids.push_back(id);
        else if (node.type == "DW") dry_waterhole_ids.push_back(id);
    }

    // Load edges
    for (auto& edge_data : network_data["adjacency"]) {
        Edge e;
        e.u = edge_data["u"];
        e.v = edge_data["v"];
        e.weight = edge_data["weight"];
        edges.push_back(e);
    }

    // Load grid
    for (auto& [key, val] : grid_data["grid"].items()) {
        string keystr = key;
        int comma = keystr.find(',');
```

```
        int c = stoi(keystr.substr(0, comma));
        int r = stoi(keystr.substr(comma + 1));

        GridCell cell;
        cell.biome = val[0];
        if (!val[1].is_null()) {
            cell.modifier = val[1];
        } else {
            cell.modifier = "";
        }
        grid[{c, r}] = cell;
    }
    //waterhole grid
    for(int wh_id:waterhole_ids){
        for(auto&node:nodes){
            if(node.id==wh_id){
                int col = (int)(node.x / 5.0);
                int row = (int)(node.y / 5.0);
                col = max(0, min(col, cols-1));
                row = max(0, min(row, rows-1));
                waterhole_grid[wh_id] = {row, col};
                break;
            }
        }
    }
    //dw grid
    for(int dw_id : dry_waterhole_ids){
        for(auto& node : nodes){
            if(node.id == dw_id){
                int col = (int)(node.x / 5.0);
                int row = (int)(node.y / 5.0);
                col = max(0, min(col, cols-1));
                row = max(0, min(row, rows-1));
                dry_waterhole_grid[dw_id] = {row, col};
                break;
            }
        }
    }
}

void computeCells() {
    // 1. Build adjacency
    map<int, vector<pair<int, double>>> adj;
    for (auto& e : edges) {
        adj[e.u].push_back({e.v, e.weight});
        adj[e.v].push_back({e.u, e.weight});
    }
    // 2. Dijkstra from each gate
    map<int, float> gate_dist;

    for (int gate_id : gate_ids) {
        map<int, double> dist;
        priority_queue<pair<double, int>, vector<pair<double, int>>, greater<>> pq;

        dist[gate_id] = 0;
        pq.push({0, gate_id});

        while (!pq.empty()) {
            auto [d, u] = pq.top(); pq.pop();
            if (d > dist[u]) continue;

            for (auto [v, w] : adj[u]) {
```

```
                if (!dist.count(v) || d + w < dist[v]) {
                    dist[v] = d + w;
                    pq.push({dist[v], v});
                }
            }
        }
        // Update gate_dist with minimum across all gates
        for (auto& [node_id, d] : dist) {
            if (!gate_dist.count(node_id) || d < gate_dist[node_id]) {
                gate_dist[node_id] = d;
            }
        }
    }
    // 3. For each grid cell, find nearest node
    vector<vector<float>> cell_gate_dist(rows, vector<float>(cols, 1e9));
    for (int r = 0; r < rows; r++) {
        for (int c = 0; c < cols; c++) {
            // Cell center coordinates (km)
            double cell_x = (c + 0.5) * 5.0;
            double cell_y = (r + 0.5) * 5.0;

            // Find nearest node
            float min_node_dist = 1e9;
            int best_node = -1;

            for (auto& node : nodes) {
                float dx = cell_x - node.x;
                float dy = cell_y - node.y;
                float euc_dist = sqrt(dx*dx + dy*dy);

                if (euc_dist < min_node_dist) {
                    min_node_dist = euc_dist;
                    best_node = node.id;
                }
            }

            if (best_node != -1 && gate_dist.count(best_node)) {
                // Off-road times 2 + road distance
                cell_gate_dist[r][c] = min_node_dist * 2 + gate_dist[best_node];
            }
        }
    }
}
void createFences(){
    fences.clear();

    for(int r=0;r<rows;r++){
        for(int c=0;c<cols;c++){
            auto cell=grid[{c,r}];

            if(cell.modifier=="Walled"){
                FenceSegment seg;
                seg.grid_r=r;
                seg.grid_c=c;
                for(auto&b : biomes){
                    if(b.code==cell.biome){
                        seg.lambda_k=b.lambda_base;
                        seg.w_k=b.w_base;
                        break;
                    }
                }
                fences.push_back(seg);
```

```
            }
        }
    }
    //normalize w_k
    float w_sum=0;
    for(auto&seg:fences) w_sum+=seg.w_k;
    if(w_sum>0){
        for(auto&seg:fences) seg.w_k/=w_sum;
    }
    total_fence_km = fences.size() * 5.0f;  // each segment 5km
}

void computeCellPoachingRisk() {
    cell_poach_risk.assign(rows, vector<float>(cols, 0));

    for (int r = 0; r < rows; r++) {
        for (int c = 0; c < cols; c++) {
            auto cell = grid[{c, r}];
            // Biome base risk
            float risk = 0;
            for (auto& b : biomes) {
                if (b.code == cell.biome) {
                    risk = b.poach_risk;
                    break;
                }
            }
            // Waterhole presence
            bool has_waterhole = false;
            for (int wh_id : waterhole_ids) {
                auto it = waterhole_grid.find(wh_id);
                if (it != waterhole_grid.end() && it->second.first == r &&
                it->second.second == c) {
                    has_waterhole = true;
                    break;
                }
            }
            if (has_waterhole) risk += 0.8f;

            cell_poach_risk[r][c] = max(0.0f, min(1.0f, risk));
        }
    }
}

void computeCellFireRisk() {
    cell_fire_risk.assign(rows, vector<float>(cols, 0));
    int match_count = 0;
    int total_flammable = 0;

    for (int r = 0; r < rows; r++) {
        for (int c = 0; c < cols; c++) {
            auto cell = grid[{c, r}];
            // Skip non-flammable biomes
            if (cell.biome == "Pan" || cell.biome == "Outside") {
                cell_fire_risk[r][c] = 0;
                continue;
            }
            total_flammable++;

            // Biome base fire risk
            float risk = 0;
            bool found = false;
            for (auto& b : biomes) {
```

```
                // Convert grid biome name to code
                string biome_code;
                if (cell.biome == "Pan") biome_code = "P";
                else if (cell.biome == "Grass Savanna") biome_code = "GS";
                else if (cell.biome == "Shrub Savanna") biome_code = "SS";
                else if (cell.biome == "Tree Savanna") biome_code = "TS";
                else if (cell.biome == "Outside") biome_code = "O";
                else continue;

                if (b.code == biome_code) {
                    risk = b.fire_risk;
                    found = true;
                    match_count++;
                    break;
                }
            }

            if (!found) {
                cout << "No match for biome: " << cell.biome << " at
                (" << r << "," << c << ")" << endl;
                continue;
            }

            // Check for waterhole presence
            bool has_waterhole = false;
            for (int wh_id : waterhole_ids) {
                auto it = waterhole_grid.find(wh_id);
                if (it != waterhole_grid.end() && it->second.first == r
                && it->second.second == c) {
                    has_waterhole = true;
                    break;
                }
            }
            if (has_waterhole) risk -= 0.2f;

            // Check for dry waterhole presence
            bool has_dry = false;
            for (int dw_id : dry_waterhole_ids) {
                auto it = dry_waterhole_grid.find(dw_id);
                if (it != dry_waterhole_grid.end() && it->second.first == r
                && it->second.second == c) {
                    has_dry = true;
                    break;
                }
            }
            if (has_dry) risk += 0.4f;

            cell_fire_risk[r][c] = max(0.0f, min(1.0f, risk));
        }
    }
    cout << "Matched " << match_count << " out of " << total_flammable
    << " flammable cells" << endl;
}

void computeCampCoverage() {
    camp_cells.clear();

    // Get camp grid positions
    for (int camp_id : camp_ids) {
        for (auto& node : nodes) {
            if (node.id == camp_id) {
                int r = (int)(node.y / 5.0);
```

```
                int c = (int)(node.x / 5.0);
                r = max(0, min(r, rows-1));
                c = max(0, min(c, cols-1));
                camp_cells.push_back({r, c});
                break;
            }
        }
    }

    // Precompute patrol coverage (3x3 with falloff)
    camp_patrol_coverage.assign(rows, vector<float>(cols, 0));
    for (auto [cr, cc] : camp_cells) {
        for (int dr = -1; dr <= 1; dr++) {
            for (int dc = -1; dc <= 1; dc++) {
                int nr = cr + dr;
                int nc = cc + dc;
                if (nr >= 0 && nr < rows && nc >= 0 && nc < cols) {
                    float w = weights.pw_3[dr+1][dc+1];
                    camp_patrol_coverage[nr][nc] += pow(w, p_params.CAMP_PATROL_EFF);
                }
            }
        }
    }

    // Precompute fire coverage (5x5 with falloff)
    camp_fire_coverage.assign(rows, vector<float>(cols, 0));
    int rad = f_params.CAMP_F_RADIUS;
    for (auto [cr, cc] : camp_cells) {
        for (int dr = -rad; dr <= rad; dr++) {
            for (int dc = -rad; dc <= rad; dc++) {
                int nr = cr + dr;
                int nc = cc + dc;
                if (nr >= 0 && nr < rows && nc >= 0 && nc < cols) {
                    float w = weights.fw_5[dr+rad][dc+rad];
                    camp_fire_coverage[nr][nc] = max(camp_fire_coverage[nr][nc],
                    0.7f * w);
                }
            }
        }
    }
    // After computing camp_fire_coverage
    float max_val = 0;
    for (int r = 0; r < rows; r++) {
        for (int c = 0; c < cols; c++) {
            if (camp_fire_coverage[r][c] > max_val) {
                max_val = camp_fire_coverage[r][c];
            }
        }
    }
    cout << "Max camp fire coverage: " << max_val << endl;
}

void computeLookoutCoverage() {
    lookout_cells.clear();

    // Get lookout grid positions
    for (int lookout_id : lookout_ids) {
        for (auto& node : nodes) {
            if (node.id == lookout_id) {
                int r = (int)(node.y / 5.0);
                int c = (int)(node.x / 5.0);
                r = max(0, min(r, rows-1));
```

```
                c = max(0, min(c, cols-1));
                lookout_cells.push_back({r, c});
                break;
            }
        }
    }

    // Precompute coverage (5x5 with same weights as ranger patrol)
    lookout_coverage.assign(rows, vector<float>(cols, 0));
    int rad = f_params.CAMP_F_RADIUS;  // using same radius as camps (5x5)

    for (auto [lr, lc] : lookout_cells) {
        for (int dr = -rad; dr <= rad; dr++) {
            for (int dc = -rad; dc <= rad; dc++) {
                int nr = lr + dr;
                int nc = lc + dc;
                if (nr >= 0 && nr < rows && nc >= 0 && nc < cols) {
                    float w = weights.fw_5[dr+rad][dc+rad];
                    lookout_coverage[nr][nc] = max(lookout_coverage[nr][nc], w);
                }
            }
        }
    }
    // After computing lookout_coverage
    float max_lookout = 0;
    for (int r = 0; r < rows; r++) {
        for (int c = 0; c < cols; c++) {
            if (lookout_coverage[r][c] > max_lookout) {
                max_lookout = lookout_coverage[r][c];
            }
        }
    }
    cout << "Max lookout coverage: " << max_lookout << endl;
}
```

```
#include "Params.h"
#include <string>
#include <vector>
#include <iostream>
using namespace std;
vector<BiomeData> biomes;
ModifierData mod_data;
PoachingParams p_params;
FireParams f_params;
BarrierParams b_params;
ResourceConstraints cons;
Weights weights;

void setBiomes(){
    BiomeData P, GS, SS, TS, O;
    // Pan
    P.code = "P";
    P.resistance = 3.5;
    P.fire_risk = 0.0;
    P.poach_risk = 0.1;
    P.lambda_base = 1.0;
    P.w_base = 0.1;
    // Grass Savanna
    GS.code = "GS";
    GS.resistance = 1.2;
    GS.fire_risk = 0.4;
    GS.poach_risk = 0.4;
```

```
    GS.lambda_base = 1.5;
    GS.w_base = 0.4;
    // Shrub Savanna
    SS.code = "SS";
    SS.resistance = 1.5;
    SS.fire_risk = 0.7;
    SS.poach_risk = 0.7;
    SS.lambda_base = 2.0;
    SS.w_base = 0.7;
    // Tree Savanna
    TS.code = "TS";
    TS.resistance = 2.0;
    TS.fire_risk = 1.0;
    TS.poach_risk = 1.0;
    TS.lambda_base = 3.0;
    TS.w_base = 1.0;
    // Outside
    O.code = "O";
    O.resistance = 1.0;
    O.fire_risk = 0.0;
    O.poach_risk = 0.0;
    // O doesn't need lambda_base or w_base (not fence)
    biomes={P,GS,SS,TS,O};
}

void setMods(){
    ModifierData R, W, WH, DH;

    // Roaded
    R.code = "R";
    R.resistance = 0.7;
    R.fire_risk = 0.0;
    R.poach_risk = 0.3;
    // Walled
    W.code = "W";
    W.resistance = 1.0;
    W.fire_risk = 0.0;
    W.poach_risk = -0.2;
    // Waterhole (d?)
    WH.code = "WH";
    WH.resistance = 1.0;
    WH.fire_risk = -0.2;
    WH.poach_risk = 0.8;
    // Dry Waterhole
    DH.code = "DH";
    DH.resistance = 1.0;
    DH.fire_risk = 0.3;
    DH.poach_risk = -0.3;
    vector<ModifierData> modifiers = {R, W, WH, DH};
}

void setPoach(){
    p_params.T_DET_0 = 6.0;
    p_params.KAPPA = 1.5;
    p_params.SIGMA_DET = 0.8;
    p_params.PATROL_K = 2.0;
    p_params.RANGER_P_RADIUS = 1;
    p_params.GATE_D_WEIGHT = 0.02;
    p_params.ROAD_D_WEIGHT = 0.03;
    p_params.P_ADJ_WEIGHT = 0.8;
    p_params.P_DIAG_WEIGHT = 0.6;
    p_params.CAMP_PATROL_EFF=2.0;
```

```
    p_params.UAV_D_RADIUS=1;
}

void setFire(){
    f_params.TAU = 6.0;
    f_params.RANGER_F_RADIUS = 2;
    f_params.CAMP_F_RADIUS = 2;
    f_params.UAV_F_RADIUS = 3;
    f_params.SAT_BASE = 0.25;
    f_params.F_ADJ_WEIGHT = 0.9;
    f_params.F_DIAG_WEIGHT = 0.8;
    f_params.PATROL_K = 2.0;
}

void setBarrier(){
    b_params.PATROL_GROUP_SIZE = 4;
    b_params.PATROL_SPEED = 2.0;
    b_params.PATROL_TIME = 8.0;
    b_params.BASE_TIME = 4.0;
    b_params.TOTAL_TIME = 720;
}

void setCons(){
    cons.H_MAX = 90;
    cons.B_MAX = 1200;
    cons.C_CAM = 8;
    cons.C_UAV = 5;
    cons.C_RANGER = 2;
    cons.C_VOL = 0.4;
    cons.WH_CAM_MAX = 3;
}

void setWeights(){
    weights.ALPHA = 20.0/41.0;  // 0.488
    weights.BETA = 12.0/41.0;   // 0.293
    weights.GAMMA = 9.0/41.0;   // 0.220
    weights.pw_3 = {
        {p_params.P_DIAG_WEIGHT, p_params.P_ADJ_WEIGHT, p_params.P_DIAG_WEIGHT},
        {p_params.P_ADJ_WEIGHT, 1.0f, p_params.P_ADJ_WEIGHT},
        {p_params.P_DIAG_WEIGHT, p_params.P_ADJ_WEIGHT, p_params.P_DIAG_WEIGHT}
    };

    weights.pw_5 = {
        {p_params.P_DIAG_WEIGHT, p_params.P_ADJ_WEIGHT, p_params.P_ADJ_WEIGHT,
        p_params.P_ADJ_WEIGHT, p_params.P_DIAG_WEIGHT},
        {p_params.P_ADJ_WEIGHT, 1.0f, 1.0f, 1.0f, p_params.P_ADJ_WEIGHT},
        {p_params.P_ADJ_WEIGHT, 1.0f, 1.0f, 1.0f, p_params.P_ADJ_WEIGHT},
        {p_params.P_ADJ_WEIGHT, 1.0f, 1.0f, 1.0f, p_params.P_ADJ_WEIGHT},
        {p_params.P_DIAG_WEIGHT,
        p_params.P_ADJ_WEIGHT, p_params.P_ADJ_WEIGHT, p_params.P_ADJ_WEIGHT,
        p_params.P_DIAG_WEIGHT}
    };
    weights.fw_5 = {
        {f_params.F_DIAG_WEIGHT, f_params.F_ADJ_WEIGHT,
        f_params.F_ADJ_WEIGHT, f_params.F_ADJ_WEIGHT, f_params.F_DIAG_WEIGHT},
        {f_params.F_ADJ_WEIGHT, 1.0f, 1.0f, 1.0f, f_params.F_ADJ_WEIGHT},
        {f_params.F_ADJ_WEIGHT, 1.0f, 1.0f, 1.0f, f_params.F_ADJ_WEIGHT},
        {f_params.F_ADJ_WEIGHT, 1.0f, 1.0f, 1.0f, f_params.F_ADJ_WEIGHT},
        {f_params.F_DIAG_WEIGHT, f_params.F_ADJ_WEIGHT, f_params.F_ADJ_WEIGHT,
        f_params.F_ADJ_WEIGHT,
        f_params.F_DIAG_WEIGHT}
    };
```

```
    weights.fw_7 = {
        {f_params.F_DIAG_WEIGHT, f_params.F_ADJ_WEIGHT, f_params.F_ADJ_WEIGHT,
        f_params.F_ADJ_WEIGHT, f_params.F_ADJ_WEIGHT, f_params.F_ADJ_WEIGHT,
        f_params.F_DIAG_WEIGHT},
        {f_params.F_ADJ_WEIGHT, 1.0f, 1.0f, 1.0f, 1.0f, 1.0f, f_params.F_ADJ_WEIGHT},
        {f_params.F_ADJ_WEIGHT, 1.0f, 1.0f, 1.0f, 1.0f, 1.0f, f_params.F_ADJ_WEIGHT},
        {f_params.F_ADJ_WEIGHT, 1.0f, 1.0f, 1.0f, 1.0f, 1.0f, f_params.F_ADJ_WEIGHT},
        {f_params.F_ADJ_WEIGHT, 1.0f, 1.0f, 1.0f, 1.0f, 1.0f, f_params.F_ADJ_WEIGHT},
        {f_params.F_ADJ_WEIGHT, 1.0f, 1.0f, 1.0f, 1.0f, 1.0f, f_params.F_ADJ_WEIGHT},
        {f_params.F_DIAG_WEIGHT, f_params.F_ADJ_WEIGHT, f_params.F_ADJ_WEIGHT,
        f_params.F_ADJ_WEIGHT, f_params.F_ADJ_WEIGHT, _params.F_ADJ_WEIGHT,
        f_params.F_DIAG_WEIGHT}
    };
}

void setAll(){
    setBiomes();
    setMods();
    setPoach();
    setFire();
    setBarrier();
    setCons();
    setWeights();
}
```

```
#include "DataLoader.h"
#include "Params.h"
#include "Greedy.h"
#include "Calculations.h"
#include <iostream>
#include <iomanip>
#include <fstream>

using namespace std;

int main()
{
    cout << "=== Loading Data ===" << endl;
    loadAllData();

    cout << "=== Computing Cells ===" << endl;
    computeCells();

    cout << "=== Creating Fences ===" << endl;
    createFences();

    cout << "=== Setting Parameters ===" << endl;
    setAll();

    // Debug info
    cout << "Camps found: " << camp_ids.size() << endl;
    cout << "Lookouts found: " << lookout_ids.size() << endl;
    cout << "Waterholes found: " << waterhole_ids.size() << endl;
    cout << "Fence segments: " << fences.size() << endl;

    cout << "=== Computing Cell Risks ===" << endl;
    computeCellPoachingRisk();
    computeCellFireRisk();

    cout << "=== Computing Coverage ===" << endl;
    computeCampCoverage();
    computeLookoutCoverage();
```

```
cout << "=== Initializing Solution ===" << endl;
Solution sol = initSolution();

// Compute initial indices using the solution (which has zero resources)
float api = computeAPI(sol);
float hpi = computeHPI(sol);
float bii = computeBII(sol.volunteers);
float f = weights.ALPHA * api + weights.BETA * hpi + weights.GAMMA * bii;
sol.current_f = f;

// Output baseline results
cout << fixed << setprecision(4);
cout << "\n=== BASELINE RESULTS (No Resources) ===" << endl;
cout << "API: " << api << endl;
cout << "HPI: " << hpi << endl;
cout << "BII: " << bii << endl;
cout << "F:   " << f << endl;

// Run greedy optimization
cout << "\n=== Running Greedy Optimization ===" << endl;
sol = runGreedy(sol);

// Output final results
cout << "\n=== FINAL RESULTS ===" << endl;
cout << "API: " << computeAPI(sol) << endl;
cout << "HPI: " << computeHPI(sol) << endl;
cout << "BII: " << computeBII(sol.volunteers) << endl;
cout << "F:   " << sol.current_f << endl;
cout << "\nResources used:" << endl;
cout << "Rangers: " << sol.total_rangers << " (" << sol.total_rangers
/ 5 << " squads)" << endl;
cout << "UAVs: " << sol.total_uav << endl;
cout << "Cameras: " << sol.total_cameras << endl;
cout << "Volunteers: " << sol.volunteers << endl;
cout << "Budget used: " << sol.budget_used << "/" << cons.R_MAX << endl;

// --- LATEX TABLE GENERATION START ---
// Calculate contributions based on history
float ranger_f_contrib = 0.0f;
float uav_f_contrib = 0.0f;
float cam_f_contrib = 0.0f;
float vol_f_contrib = 0.0f;

float ranger_cost = 0.0f;
float uav_cost = 0.0f;
float cam_cost = 0.0f;
float vol_cost = 0.0f;

int ranger_squads = 0;
int uav_count = 0;
int cam_count = 0;
int vol_teams = 0;

// Baseline contribution is the initial F score calculated before optimization
float baseline_f = f; // 'f' variable from earlier computation

float last_hist_f = baseline_f;

for (const auto &rec : sol.history)
{
    // Calculate actual delta F observed in this step
```

```
        float delta_f = rec.f_score - last_hist_f;
        last_hist_f = rec.f_score;

        // Attribute to resource type
        if (rec.placement.type == "RANGER")
        {
            ranger_f_contrib += delta_f;
            ranger_cost += 5.0f; // COST_RANGER
            ranger_squads++;
        }
        else if (rec.placement.type == "UAV")
        {
            uav_f_contrib += delta_f;
            uav_cost += 7.0f; // COST_UAV
            uav_count++;
        }
        else if (rec.placement.type == "CAMERA")
        {
            cam_f_contrib += delta_f;
            cam_cost += 0.2f; // COST_CAMERA
            cam_count++;
        }
        else if (rec.placement.type == "VOLUNTEER")
        {
            vol_f_contrib += delta_f;
            vol_cost += 2.0f; // COST_VOLUNTEER
            vol_teams++;
        }
    }

    // Prepare LaTeX Table Output
    cout << "\n=== LaTeX Output Table ===\n"
         << endl;
    cout << "\\begin{table}[!htp]" << endl;
    cout << "\\centering" << endl;
    cout << "\\caption{Optimal Resource Allocation Summary}" << endl;
    cout << "\\label{tab:resource_summary}" << endl;
    cout << "\\begin{tabular}{lcccc}" << endl;
    cout << "\\toprule" << endl;
    cout << "\\textbf{Resource Type} & \\textbf{Units Deployed} &
    \\textbf{Budget Cost} & \\textbf{$\\mathcal{F}$ Contribution}
    & \\textbf{Primary Index}
    \\\\" << endl;
    cout << "\\midrule" << endl;

    // Rangers Row
    cout << "Rangers    & " << (ranger_squads * 5) << " (" << ranger_squads << " squads)
    & "<< fixed << setprecision(1) << ranger_cost << "  & "
         << fixed << setprecision(3)
         << ranger_f_contrib << " & API \\\\" << endl;

    // UAVs Row
    cout << "UAVs       & " << uav_count << "               & "
         << fixed << setprecision(1) << uav_cost << "   & "
         << fixed << setprecision(3)
         << uav_f_contrib << " & HPI/API \\\\" << endl;

    // Cameras Row
    cout << "AI Cameras & " << cam_count << "                & "
         << fixed << setprecision(1) << cam_cost << "  & "
         << fixed << setprecision(3)
         << cam_f_contrib << " & API \\\\" << endl;
```

```
    // Volunteers Row
    cout << "Volunteers & " << (vol_teams * 6) << " (" << vol_teams << " teams)
& "
         << fixed << setprecision(1) << vol_cost << "   & "
         << fixed << setprecision(3)
         << vol_f_contrib << " & BII \\\\" << endl;

    // Baseline Row
    cout << "Baseline & --- & 0 & " << fixed << setprecision(3) << baseline_f << "
    & --- \\\\" << endl;

    cout << "\\midrule" << endl;

    // Total Row
    cout << "\\textbf{Total} & --- & \\textbf{" << fixed << setprecision(1)
    << sol.budget_used << "} & \\textbf{"
         << fixed << setprecision(3) << sol.current_f << "} & --- \\\\" << endl;

    cout << "\\bottomrule" << endl;
    cout << "\\end{tabular}" << endl;
    cout << "\\end{table}" << endl;
    // --- LATEX TABLE GENERATION END ---

    // Save history to file for visualization
    ofstream out("his.csv");
    out << "step,f_score,api,hpi,bii,type,r,c,waterhole_id,gain,budget,rangers,
    uav,cameras,volunteers\n";
    for (const auto &rec : sol.history)
    {
        out << rec.step << ","
            << rec.f_score << ","
            << rec.api << ","
            << rec.hpi << ","
            << rec.bii << ","
            << rec.placement.type << ","
            << rec.placement.r << ","
            << rec.placement.c << ","
            << rec.placement.waterhole_id << ","
            << rec.placement.gain << ","
            << rec.budget_used << ","
            << rec.total_rangers << ","
            << rec.total_uav << ","
            << rec.total_cameras << ","
            << rec.volunteers << "\n";
    }
    out.close();
    cout << "\nHistory saved to greedy_history.csv" << endl;

    cout << "\n=== Initialization Complete ===" << endl;
    cout << "Rows: " << rows << ", Cols: " << cols << endl;
    cout << "Waterholes: " << waterhole_ids.size() << endl;
    cout << "Fence segments: " << fences.size() << endl;
    cout << "Total fence km: " << total_fence_km << endl;

    return 0;
}
```

Python Visualization

```
import json
```

```
import csv
import os
from collections import defaultdict
import matplotlib.pyplot as plt
import matplotlib.patches as patches
import matplotlib.lines as mlines

# -------------------- 1. Advanced Academic Color Configuration (HEX) --------------------
# Reduced saturation, increased transparency, ensuring the base map is clearly visible
BIOMES_COLORS = {
    "Pan": "#EAE0C8",           # Light Beige
    "Grass Savanna": "#C3D8B4", # Soft Light Green
    "Shrub Savanna": "#7FB069", # Olive Green
    "Tree Savanna": "#3A7D44",  # Dark Forest Green
    "Outside": "#34495E"        # Dark Blue Grey
}
BIOME_ALPHA = 0.35              # Increase transparency, do not obscure the base map

MODIFIER_COLORS = {
    "Road": "#E67E22",          # Warm Amber Orange (replacing harsh bright yellow)
    "Walled": "#8E44AD"         # Calm Dark Purple (replacing harsh neon pink)
}

NODE_TYPES = {
    "WH": {"name": "Waterhole", "color": "#3498DB"},      # Blue
    "DW": {"name": "Dry Waterhole", "color": "#A04000"},  # Brown
    "LO": {"name": "Lookout", "color": "#F1C40F"},        # Yellow
    "MC": {"name": "Main Camp", "color": "#27AE60"},      # Green
    "RI": {"name": "Road Intersect", "color": "#95A5A6"}, # Grey
    "GA": {"name": "Gate", "color": "#C0392B"}            # Dark Red
}

# Deployment area colors (consistent with your previous question's color scheme)
RANGER_COLOR = "#2ecc71"
UAV_COLOR = "#3498db"

# -------------------- 2. File Path Configuration --------------------
MAP_IMAGE = "Etosha-Map-2025.jpg"
CONFIG_JSON = "map_config.json"
CALIB_JSON = "calibration.json"
GRID_DATA_FILE = "final_grid_data.json"
NODE_DATA_FILE = "park_data.json"
CSV_FILE = "his.csv"

# -------------------- 3. Load Data --------------------
with open(CALIB_JSON, "r") as f:
    calib = json.load(f)
PX_PER_KM = calib["px_per_km"]
KM_5_PX = PX_PER_KM * 5

# Load grid
grid_data = {}
if os.path.exists(GRID_DATA_FILE):
    with open(GRID_DATA_FILE, "r") as f:
        saved = json.load(f)
        for k, v in saved.get("grid", {}).items():
            c, r = map(int, k.split(','))
            grid_data[(c, r)] = v

# Load nodes and edges
nodes, edges = [], []
if os.path.exists(NODE_DATA_FILE):
```

```
    with open(NODE_DATA_FILE, "r") as f:
        d = json.load(f)
        nodes = d.get('nodes', [])
        edges = d.get('edges', [])

# Load deployment history
ranger_counts = defaultdict(int)
uav_counts = defaultdict(int)
camera_counts = defaultdict(int)
if os.path.exists(CSV_FILE):
    with open(CSV_FILE, "r") as f:
        for row in csv.DictReader(f):
            c, r = int(row["c"]), int(row["r"])
            if row["type"] == "RANGER":
                ranger_counts[(c, r)] += 1
            elif row["type"] == "UAV":
                uav_counts[(c, r)] += 1
            elif row["type"] == "CAMERA":
                camera_counts[(c, r)] += 1

# -------------------- 4. Start Plotting with Matplotlib --------------------
# Set canvas size (width 16, height proportional)
img_raw = plt.imread(MAP_IMAGE)
img_h, img_w = img_raw.shape[:2]
fig, ax = plt.subplots(figsize=(16, 16 * (img_h / img_w)))

# Show original image as background
ax.imshow(img_raw, extent=[0, img_w, img_h, 0])

# 4.1 Draw Biome Grid (Semi-transparent squares)
for (c, r), layers in grid_data.items():
    biome_name = layers[0]
    if biome_name in BIOMES_COLORS:
        wx, wy = c * KM_5_PX, r * KM_5_PX
        rect = patches.Rectangle((wx, wy), KM_5_PX, KM_5_PX,
                                 linewidth=0, facecolor=BIOMES_COLORS[biome_name],
                                 alpha=BIOME_ALPHA, zorder=1)
        ax.add_patch(rect)

# 4.2 Draw Boundaries and Grid Roads (Modifiers)
for (c, r), layers in grid_data.items():
    mod_name = layers[1]
    if mod_name in MODIFIER_COLORS:
        cx, cy = (c + 0.5) * KM_5_PX, (r + 0.5) * KM_5_PX
        color = MODIFIER_COLORS[mod_name]

        # Draw grid center point
        ax.scatter(cx, cy, color=color, s=20, zorder=2)

        # Connection logic (right and down, avoid duplicate drawing)
        for dc, dr in [(1, 0), (0, 1), (1, 1), (-1, 1)]:
            nb = (c + dc, r + dr)
            if nb in grid_data and grid_data[nb][1] == mod_name:
                # Avoid diagonal crossing when orthogonal exists
                if abs(dc) == 1 and abs(dr) == 1:
                    if grid_data.get((c+dc, r), [None, None])[1] == mod_name or \
                       grid_data.get((c, r+dr), [None, None])[1] == mod_name:
                        continue
                nb_cx, nb_cy = (nb[0] + 0.5) * KM_5_PX, (nb[1] + 0.5) * KM_5_PX
                ax.plot([cx, nb_cx], [cy, nb_cy], color=color, linewidth=2, alpha=0.8,
                zorder=2)
```

```
# 4.3 Draw Road Network (Edges)
for e in edges:
    u, v = e['u'], e['v']
    if u < len(nodes) and v < len(nodes):
        x1, y1 = nodes[u]['pix']
        x2, y2 = nodes[v]['pix']
        color = "#95A5A6" if e.get('is_auto', False) else "#E74C3C"
        ax.plot([x1, x2], [y1, y2], color=color, linewidth=2.5, alpha=0.9, zorder=3)

# 4.4 Draw Nodes
for n in nodes:
    x, y = n['pix']
    typ = n.get('type', 'RI')
    color = NODE_TYPES.get(typ, NODE_TYPES['RI'])['color']
    # Add a white border (edgecolors) to make nodes stand out in complex background
    ax.scatter(x, y, color=color, s=80, edgecolors='white', linewidth=1.5, zorder=4)

# 4.5 Draw Deployment Areas (Rangers & UAVs)
def draw_coverage(c, r, size, color, alpha, zorder):
    half = size // 2
    left = (c - half) * KM_5_PX
    top = (r - half) * KM_5_PX
    width = size * KM_5_PX
    rect = patches.Rectangle((left, top), width, width,
                             linewidth=1, edgecolor=color, facecolor=color,
                             alpha=alpha, zorder=zorder)
    ax.add_patch(rect)

for (c, r), count in ranger_counts.items():
    draw_coverage(c, r, 5, RANGER_COLOR, 0.1, zorder=5)  # 5x5 Outer circle light color
    draw_coverage(c, r, 3, RANGER_COLOR, 0.2, zorder=5)  # 3x3 Inner circle dark color
    cx, cy = (c + 0.5) * KM_5_PX, (r + 0.5) * KM_5_PX
    ax.scatter(cx, cy, color=RANGER_COLOR, marker='*', s=150, edgecolors='white',
    zorder=6, label='Ranger' if 'Ranger' not in ax.get_legend_handles_labels()[1] else "")

if uav_counts:
    uav_cols = [k[0] for k in uav_counts.keys()]

    min_uav_c = min(uav_cols)
    max_uav_c = max(uav_cols)
else:
    min_uav_c = max_uav_c = -1

for (c, r), count in uav_counts.items():
    # if c == min_uav_c or c == max_uav_c:
    #     continue
    draw_coverage(c, r, 7, UAV_COLOR, 0.08, zorder=5)
    draw_coverage(c, r, 5, UAV_COLOR, 0.15, zorder=5)
    cx, cy = (c + 0.5) * KM_5_PX, (r + 0.5) * KM_5_PX
    ax.scatter(cx, cy, color=UAV_COLOR, marker='^', s=120, edgecolors='white', zorder=6,
    label='UAV' if 'UAV' not in ax.get_legend_handles_labels()[1] else "")

for (c, r), count in camera_counts.items():
    cx, cy = (c + 0.5) * KM_5_PX, (r + 0.5) * KM_5_PX
    ax.scatter(cx, cy, color='green', marker='s', s=50, alpha=0.7, edgecolors='white',
    zorder=6, label='Camera' if 'Camera' not in ax.get_legend_handles_labels()[1] else "")

# ====Draw Volunteer Outposts (Evenly distributed on boundaries)
# ============================================================================
# 1. Extract all boundary cells
walled_cells = [(c, r) for (c, r), layers in grid_data.items() if layers[1] == 'Walled']
```

```
if walled_cells:
    # 2. Greedy algorithm: Connect disordered boundary cells end-to-end to form a circle
    ordered_walls = [walled_cells.pop(0)]
    while walled_cells:
        last = ordered_walls[-1]
        # Find the next boundary point closest to the current point
        nearest_idx = min(range(len(walled_cells)),
                          key=lambda i: (walled_cells[i][0]-last[0])**2
                          + (walled_cells[i][1]-last[1])**2)
        ordered_walls.append(walled_cells.pop(nearest_idx))

    # 3. Evenly extract 15 points along the boundary line
    num_outposts = 15
    step = len(ordered_walls) / num_outposts
    outpost_cells = [ordered_walls[int(i * step)] for i in range(num_outposts)]

    # 4. Convert to canvas coordinates (center point)
    outpost_x = [(c + 0.5) * KM_5_PX for c, r in outpost_cells]
    outpost_y = [(r + 0.5) * KM_5_PX for c, r in outpost_cells]


    ax.scatter(outpost_x, outpost_y, color='#FF9F1C', marker='P', s=200,
               edgecolors='white', linewidth=1.5, zorder=7,

               label='Volunteer Team' if 'Volunteer Team' not in
               ax.get_legend_handles_labels()[1] else "")


# -------------------- 5. Legend and Format Adjustment --------------------
# Remove axis ticks
ax.set_xticks([])
ax.set_yticks([])
ax.set_frame_on(False)

# Dynamically generate legend for nodes and regions
legend_elements = [

    patches.Patch(color=RANGER_COLOR, alpha=0.5, label='Ranger Coverage'),
    patches.Patch(color=UAV_COLOR, alpha=0.5, label='UAV Coverage'),
    mlines.Line2D([], [], color='green', marker='s', linestyle='None',
    markersize=10, markeredgecolor='white', label='Camera'),
    # ---- New line below ----
    mlines.Line2D([], [], color='#FF9F1C', marker='P', linestyle='None',
    markersize=11, markeredgecolor='white', label='Volunteer Team'),
    mlines.Line2D([], [], color='#8E44AD', linewidth=2, label='Walled Boundary'),

    mlines.Line2D([], [], color='#E67E22', linewidth=2, label='Road / Patrol Path'),
]
for k, v in NODE_TYPES.items():
    legend_elements.append(mlines.Line2D([], [], color=v['color'], marker='o',
    linestyle='None', markersize=8, label=v['name']))

# Place legend (top right corner or outside)
ax.legend(handles=legend_elements, loc='upper right', bbox_to_anchor=(1.15, 1),
          fontsize=14, framealpha=0.9, title="Map Elements", title_fontsize=14)

# -------------------- 6. Save as PDF --------------------
plt.tight_layout()
output_pdf = "etosha_deployment_map.pdf"
plt.savefig(output_pdf, format='pdf', dpi=300, bbox_inches='tight')
plt.show()
```