Integrated Malaria Intervention Optimization Model

Md. Mehran Alam

December 2025

Abstract

This document provides a mathematical and computational description of an integrated malaria intervention optimization model that couples disease dynamics (SIR-type ODEs), behavioral response (belief and effort dynamics), and optimization for constrained resource allocation. The model is implemented in R. I present the equations, parameterization, optimization formulations, solver details, and include the full R implementation in the Appendix.

Contents

1 Introduction

Malaria remains a leading cause of morbidity in many settings. Control strategies (LLINs, IRS, ACT, IPT, vaccines, campaigns) interact with population behavior. The model presented here integrates:

- A compartmental disease model (S, I, R) with a mechanistic force-of-infection driven by mosquito density and biting rates.
- Behavioral dynamics tracking belief ($B(t)$) and prevention effort ($E(t)$) that feed back into transmission.
- Intervention effect modules that modify epidemiological parameters and behavioral drivers depending on coverage.
- An optimization module using an **Integer Linear Program (ILP)** to find the globally optimal selection of interventions under budget constraints. The former greedy heuristics are retained for comparison purposes.

This document describes the final model in detail and provides runnable R code in the appendix.

2 Mathematical Model

2.1 State variables and notation

For each (representative) district we model continuous-time dynamics over $t \in [0, T]$:

$$\begin{aligned} S(t) &= \text{Susceptible proportion}, \\ I(t) &= \text{Infected proportion}, \\ R(t) &= \text{Recovered (temporarily immune) proportion}, \\ B(t) &= \text{Belief about malaria risk} \in [0,1], \\ E(t) &= \text{Prevention effort} \in [0,1]. \end{aligned}$$

Population constraint (human-only compartments):

$$S(t) + I(t) + R(t) = 1 \quad \text{(per-district proportion)}.$$

Model parameters use the notation consistent with the R implementation (subscripts "u" denote baseline untreated values).

2.2 Force of infection

The force of infection $\lambda(t)$ implemented in the code is:

$$\lambda(t) \;=\; \frac{m\, a^2\, b\, c\, e^{-\mu\tau}\, I(t)}{\mu + acI(t)} \cdot \big(1 - E(t)\big), \tag{1}$$

where

- m = mosquitoes per human (region-dependent),
- a = biting rate (bites per mosquito per day),
- b = mosquito-to-human transmission probability,

- c = human-to-mosquito transmission probability,
- μ = mosquito mortality (per day),
- τ = extrinsic incubation period (days),
- $E(t)$ = prevention effort (reduces transmission multiplicatively).

The a^2 reflects the two-step transmission (infected human $\rightarrow$ mosquito $\rightarrow$ susceptible human); $\exp(-\mu\tau)$ is survival through incubation; denominator accounts for mosquito saturation.

2.3 Disease ODEs

The coupled disease–behavior ODEs used in simulation:

$$\frac{dS}{dt} = -\lambda(t)\, S + \rho\, R, \tag{2}$$

$$\frac{dI}{dt} = \lambda(t)\, S - \gamma\, I - \delta\, I, \tag{3}$$

$$\frac{dR}{dt} = \gamma\, I - \rho\, R, \tag{4}$$

with γ = recovery rate, δ = disease mortality (small baseline), ρ = loss of immunity (return to susceptible).

2.4 Behavioral dynamics

Belief evolves via social learning, own-experience, and campaign effects, with decay:

$$\frac{dB}{dt} \;=\; \alpha_S(1-I)I(1-B) \;+\; \alpha_O I(1-B) \;+\; \alpha_C(1+\alpha_{C,boost})C(t)(1-B) \;-\; \delta_B B, \tag{5}$$

where $C(t)$ is campaign intensity (0–1) and $\alpha_{C,boost}$ scales campaign efficacy (depends on coverage in implementation).

Prevention effort is a deterministic function of belief and infection prevalence:

$$E(t) \;=\; \min\Big(1,\; \max\big(0,\; \theta_B B(t) + \theta_I I(t) + \theta_{int} - \theta_C\, c_{effort}\big)\Big), \tag{6}$$

where θ_{int} is an intervention-induced effort boost (depends on intervention coverage).

2.5 Intervention effects (coverage scaling)

Each intervention j modifies baseline parameters (examples):

- **LLIN:** reduces biting a and mosquito density m; adds behavioral boost θ_{int}.
- **IRS:** reduces m; adds behavioral boost.
- **IPT:** reduces b (susceptibility).
- **ACT:** increases γ (faster recovery).
- **Vaccine:** reduces b, increases γ.
- **Campaign:** increases α_C (learning) and directly increases θ_{int} and $C(t)$.

Coverage $\kappa \in \{0.2, 0.4, 0.6, 0.8\}$ scales each effect using convex combinations in the code:

$$\theta_{\text{eff}} = (1 - \kappa)\, \theta_{\text{u}} + \kappa\, \theta_{\text{treated}}$$

or multiplicative reductions as specified in the intervention module in the R implementation.

3 Outcome metrics and cost-effectiveness

3.1 Person-days of infection

For a simulation horizon $[0, T]$ with population N (per-district):

$$\text{PD} \;=\; N \int_0^T I(t)\, dt \quad \text{(computed numerically)}.$$

In code we sum daily $I(t)$ values and multiply by N to get person-days.

3.2 Avoided person-days

Given baseline PD (no intervention) and PD under intervention j, κ:

$$\text{APD}_{j,\kappa} \;=\; \text{PD}_{baseline} - \text{PD}_{j,\kappa}.$$

3.3 Cost function

Total cost per district for intervention j at coverage κ:

$$C_{j,\kappa,r} \;=\; c_j \cdot \kappa \cdot N \cdot \phi_r \cdot \text{SIM\_YEARS},$$

where c_j is per-person cost and ϕ_r is regional distribution multiplier (low/medium/high).

3.4 Cost-effectiveness ratio (CER)

$$\text{CER}_{j,\kappa,r} \;=\; \frac{\text{APD}_{j,\kappa,r}}{C_{j,\kappa,r}}.$$

Higher CER $\Rightarrow$ more health impact per dollar.

4 Optimization formulations

4.1 Integer Linear Program (ILP) Implemented

The primary optimization method is an exact **Integer Linear Program (ILP)**, solved using the `lpSolve` R package, which implements the exact formulation of the 0/1 Knapsack problem. This method guarantees finding the true, global optimum (maximum Avoided Person-Days).

Let decision variables $x_{r,j,\kappa} \in \{0, 1\}$ indicate selecting intervention j at coverage κ in region r.

$$\max_x \quad \sum_r \sum_j \sum_\kappa \text{APD}_{r,j,\kappa}\, x_{r,j,\kappa}$$

subject to

$$\sum_r \sum_j \sum_\kappa C_{r,j,\kappa}\, x_{r,j,\kappa} \le B.$$

Key optimization finding: The final ILP solution selected **52 actions** across 9 districts, indicating that multiple interventions can be optimally deployed in the same district (e.g., LLIN + Campaign combinations).

4.2 Greedy Heuristics (Retained for Comparison)

The greedy heuristics are no longer the primary solution but are retained for diagnostics and demonstration of the sub-optimality of approximate methods.

1. **CE-focused greedy:** Ranks all available actions by their Cost-Effectiveness Ratio (APD/C) and sequentially selects the highest-ranking actions while enforcing **one action per region** until the budget is exhausted. This method provides a fast approximation.
2. **Total-benefit (retired):** The original total-benefit greedy approach was shown to be brittle and has been replaced by the superior ILP.

5 Numerical implementation: solver details

5.1 ODE solver

The ODE system (Eqs. (2)–(5)) is solved using `lsoda` (via `deSolve::ode`) with daily time-steps by default. Typical solver settings in the R code:

- `method = "lsoda"`
- `rtol = 1e-5`, `atol = 1e-5`
- daily steps: `times = seq(0, 365, by = 1)`

The code includes robust try-catch fallback for solver failures (returns baseline metrics when the solver fails).

5.2 Parameter values

Key baseline parameter values used (see also Appendix A for full R table):

Table 1: Selected baseline parameter values (per model code)

Parameter	Symbol	Baseline value
Biting rate	a_u	0.25 (bites/mosquito/day)
Transmission (mosquito → human)	b_u	0.022
Transmission (human → mosquito)	c	0.36
Mosquito mortality	μ	0.095 day^{-1}
Extrinsic incubation	τ	10 days
Recovery untreated	γ_u	1/180 day^{-1}
Immunity waning	ρ_u	1/274 day^{-1}
Mosquito density (dry/moderate/wet)	m	5 / 20 / 35 mosquitoes/person
Population per district	N	10,000 persons
Simulation years	SIM_YEARS	5 years

Behavioral parameters (baseline):

$$\alpha_S = 0.02,\ \alpha_O = 0.01,\ \alpha_C = 0.03,\ \delta_B = 0.005,\ \theta_B = 0.6,\ \theta_I = 0.4,\ \theta_C = 0.3,\ c_{effort} = 0.5.$$

Intervention unit costs (USD/person/year) used:

$$\{\text{LLIN} = 1.33,\ \text{IRS} = 2.22,\ \text{IPT} = 1.13,\ \text{ACT} = 4.82,\ \text{vaccine} = 20.66,\ \text{campaign} = 1.50\}.$$

5.3 Simulation workflow

The main pipeline:

1. Generate action space (single interventions at multiple coverages + selected combos).
2. For each region and action, run `simulate_year()` producing final state and person-days.
3. Compute baseline PD per region (no intervention).
4. Compute avoided PD and cost for each region-action.
5. **Apply the ILP solver** to find the globally optimal allocation, and run the greedy methods for comparison.
6. Produce diagnostic tables and plots.

6 Results and diagnostics

6.1 Key Optimization Results

With a 5-year budget of $2.3 million, the model produced the following comparative results:

Table 2: Optimization method comparison (5-year horizon)

Method	Actions	Total Cost	Avoided PD (5Y)	Budget Used	PD/$1k
ILP (True Optimum)	52	$2,299,300	190,639,524	100.0%	82,912
CE-Focused (Greedy)	9	$319,200	31,954,877	13.9%	100,109

Key findings from the optimization:

- **ILP superiority:** The Integer Linear Program yields **496.6% more impact** than the greedy CE-focused approach.
- **Campaign effectiveness:** Campaigns appear in 25% of optimal interventions, contributing 26% of total impact.
- **Budget efficiency:** The ILP solution spends nearly the entire budget ($2.3M of $2.3M) for maximal impact.
- **Marginal returns:** While the greedy approach has higher CE ratio (100,109 vs 82,912 PD/$1k), the ILP achieves **5× more total avoided person-days**.
- **Combination strategies:** The optimal solution includes intervention combinations (e.g., LLIN + Campaign) in the same districts.

6.2 Campaign Impact Analysis

The behavioral component shows significant impact:

- Campaign interventions contribute **49,508,795 avoided person-days** (26% of total impact).
- Campaigns cost $652,700 (28.4% of total budget).
- Campaign cost-effectiveness: **75,852 PD/$1,000**.
- Top campaign interventions target **wet regions with low distribution costs**.

6.3 Climate-Specific Insights

Consistent with Dudley et al. (2016) but with behavioral enhancements:

- **Dry regions:** Primarily benefit from ACT-based interventions.
- **Wet regions:** Require campaigns + high coverage combinations (LLIN_campaign, ACT_campaign).
- **Distribution costs:** Medium/high costs reduce CE ratios by 20-25%.

7 Discussion

7.1 Strengths

- **Behavioral integration:** Combines mechanistic transmission with realistic behavioral feedback.
- **Optimality guarantee:** ILP provides globally optimal allocation for given inputs.
- **Flexible intervention design:** Supports combinations, coverage scaling, and campaign interactions.
- **Computational efficiency:** 333 simulations complete in 1 minute for 5-year horizon.
- **Policy relevance:** Explicitly models campaign effects and distribution costs.

7.2 Limitations

- **Aggregated districts:** Non-spatial model lacks within-district heterogeneity.
- **Deterministic dynamics:** No stochastic elements in transmission.
- **Parameter uncertainty:** No Bayesian calibration or sensitivity analysis.
- **Scale differences:** Smaller population (10,000 vs Dudley's larger scale).

7.3 Comparison with Dudley et al. (2016)

This model extends Dudley's work in several key areas:

Table 3: Model comparison with Dudley et al. (2016)

Aspect	Dudley et al. (2016)	This Model
Core dynamics	SIR (6 compartments)	**SIR + Belief + Effort**
Behavioral component	None	**Explicit belief and effort dynamics**
Campaign interventions	Not included	**Explicit campaign modeling**
Optimization	Single ILP	**ILP + Greedy comparison**
Coverage options	20%, 40%, 60%, 80%	**20%, 40%, 60%, 80%**
Combination strategies	Fixed combinations	**Dynamic combinations (LLIN_campaign, etc.)**
Results interpretation	Climate-driven only	**Climate + behavior + optimization trade-offs**

7.4 Policy Implications

- **Campaigns matter:** Information campaigns provide 26% of impact and should be included in intervention portfolios.
- **Mathematical optimization is crucial:** ILP yields 5× more impact than simple CE ranking.
- **Wet region strategy:** Require campaigns + high coverage combinations.
- **Budget allocation:** Near-full budget utilization maximizes health impact despite slightly lower CE ratios.

7.5 Planned extensions

- Multi-year dynamic budget allocation with sequencing.
- Spatial coupling across districts and travel-driven transmission.
- Parameter estimation using surveillance data (Bayesian calibration).
- Inclusion of multiple objective criteria (DALYs, equity, resistance).
- Vaccine sensitivity analysis (extending Dudley's vaccine cost analysis).

A Appendix A: Full R Implementation (Final model)

```r
# ============================================================================
# COMPLETE MALARIA INTERVENTION OPTIMIZATION MODEL - FINAL VERSION
# ============================================================================

# Load required libraries
library(deSolve)
library(ggplot2)
library(dplyr)
library(tidyr)
library(lpSolve)

# ============================================================================
# Helper: null-coalescing
# ============================================================================
`%||%` <- function(a, b) if (!is.null(a)) a else b

# ============================================================================
# 1. PARAMETER DEFINITION (Updated based on results)
# ============================================================================

params_epidemiology <- list(
  a_u = 0.25, b_u = 0.022, c = 0.36, delta = 4.7895e-5,
  gamma_u = 1/180, m_u_dry = 5, m_u_mod = 20, m_u_wet = 35,
  mu = 0.095, omega = 274, tau = 10, rho_u = 1/274, N = 10000  # Fixed: 10,000
)

params_behavior <- list(
  alpha_S = 0.02, alpha_O = 0.01, alpha_C = 0.03, delta_B = 0.005,
  theta_B = 0.6, theta_I = 0.4, theta_C = 0.3, c_effort = 0.5
)

params_interventions <- list(
  cost = c(none = 0, LLIN = 1.33, IRS = 2.22, IPT = 1.13,
           ACT = 4.82, vaccine = 20.66, campaign = 1.50),  # Fixed: 1.50
  effects = list(
    LLIN = list(a_reduction = 0.80, m_reduction = 0.64, theta_i = 0.2),
    IRS = list(m_reduction = 0.95, theta_i = 0.15),
    IPT = list(b_reduction = 0.786, theta_i = 0.1),
```

```
    ACT = list(gamma_multiplier = 18, theta_i = 0.15),
    vaccine = list(b_reduction = 0.773, gamma_multiplier = 3.27, theta_i = 0.1),
    campaign = list(alpha_C_boost = 0.4, theta_i_boost = 0.3, compliance_boost = 0.25)
  ),
  coverage_levels = c(0.2, 0.4, 0.6, 0.8)
)

regions <- expand.grid(
  climate = c("dry", "moderate", "wet"),
  dist_cost = c("low", "medium", "high"),
  stringsAsFactors = FALSE
)

initial_conditions <- list(
  dry = c(S = 0.60, I = 0.15, R = 0.25, B = 0.3, E = 0.2),
  moderate = c(S = 0.15, I = 0.15, R = 0.70, B = 0.5, E = 0.3),
  wet = c(S = 0.10, I = 0.15, R = 0.75, B = 0.4, E = 0.25)
)

SIM_YEARS <- 5
SIM_DAYS <- 365

# ============================================================================
# 2. CORE DYNAMICS FUNCTIONS
# ============================================================================

calculate_force_of_infection <- function(m, a, b, I_total, E, params) {
  numerator <- m * a^2 * b * params$c * exp(-params$mu * params$tau) * I_total
  denominator <- params$mu + a * params$c * I_total
  lambda <- numerator / denominator
  return(lambda * (1 - min(E, 1)))
}

combined_odes <- function(t, state, params) {
  S <- state["S"]; I <- state["I"]; R <- state["R"]; B <- state["B"]

  m <- params$m; a <- params$a; b <- params$b; c <- params$c
  gamma <- params$gamma; delta <- params$delta; rho <- params$rho
  mu <- params$mu; tau <- params$tau
  alpha_S <- params$alpha_S; alpha_0 <- params$alpha_0
  alpha_C <- params$alpha_C; delta_B <- params$delta_B
  theta_B <- params$theta_B; theta_I <- params$theta_I
  theta_C <- params$theta_C; c_effort <- params$c_effort
  theta_i_boost <- params$theta_i_boost %||% 0
  C_intensity <- params$C_intensity %||% 0
  alpha_C_boost <- params$alpha_C_boost %||% 0

  theta_total <- theta_B * B + theta_I * I + theta_i_boost
  E <- min(1, max(0, theta_total - theta_C * c_effort))

  lambda <- calculate_force_of_infection(m, a, b, I, E, params)

  dS <- -lambda * S + rho * R
  dI <- lambda * S - gamma * I - delta * I
  dR <- gamma * I - rho * R

  dB <- alpha_S * (1 - I) * I * (1 - B) +
    alpha_0 * I * (1 - B) +
    alpha_C * (1 + alpha_C_boost) * C_intensity * (1 - B) -
    delta_B * B

  return(list(c(dS = dS, dI = dI, dR = dR, dB = dB), E = E))
}

get_intervention_effects <- function(intervention_type, coverage, m_u, params_epi) {
  effects <- list(
    m = m_u, a = params_epi$a_u, b = params_epi$b_u,
    gamma = params_epi$gamma_u, rho = params_epi$rho_u,
    delta = params_epi$delta, mu = params_epi$mu,
    tau = params_epi$tau, c = params_epi$c,
    theta_i_boost = 0, alpha_C_boost = 0, C_intensity = 0
  )

```

```r
  if (is.null(intervention_type) || intervention_type == "none" || intervention_type == "") {
    return(effects)
  }

  interventions <- strsplit(intervention_type, "_")[[1]]

  for (int in interventions) {
    if (int %in% c("0.2", "0.4", "0.6", "0.8")) next

    if (int == "LLIN" && int %in% names(params_interventions$effects)) {
      int_effects <- params_interventions$effects[[int]]
      effects$a <- effects$a * (1 - int_effects$a_reduction * coverage)
      effects$m <- effects$m * (1 - int_effects$m_reduction * coverage)
      effects$theta_i_boost <- effects$theta_i_boost + int_effects$theta_i * coverage
    }

    if (int == "IRS" && int %in% names(params_interventions$effects)) {
      int_effects <- params_interventions$effects[[int]]
      effects$m <- effects$m * (1 - int_effects$m_reduction * coverage)
      effects$theta_i_boost <- effects$theta_i_boost + int_effects$theta_i * coverage
    }

    if (int == "IPT" && int %in% names(params_interventions$effects)) {
      int_effects <- params_interventions$effects[[int]]
      effects$b <- effects$b * (1 - int_effects$b_reduction * coverage)
      effects$theta_i_boost <- effects$theta_i_boost + int_effects$theta_i * coverage
    }

    if (int == "ACT" && int %in% names(params_interventions$effects)) {
      int_effects <- params_interventions$effects[[int]]
      current_gamma <- effects$gamma
      target_gamma <- params_epi$gamma_u * int_effects$gamma_multiplier
      effects$gamma <- (1 - coverage) * current_gamma + coverage * target_gamma
      effects$theta_i_boost <- effects$theta_i_boost + int_effects$theta_i * coverage
    }

    if (int == "vaccine" && int %in% names(params_interventions$effects)) {
      int_effects <- params_interventions$effects[[int]]
      effects$b <- effects$b * (1 - int_effects$b_reduction * coverage)
      current_gamma <- effects$gamma
      target_gamma <- params_epi$gamma_u * int_effects$gamma_multiplier
      effects$gamma <- (1 - coverage) * current_gamma + coverage * target_gamma
      effects$theta_i_boost <- effects$theta_i_boost + int_effects$theta_i * coverage
    }

    if (int == "campaign" && int %in% names(params_interventions$effects)) {
      int_effects <- params_interventions$effects[[int]]
      effects$alpha_C_boost <- int_effects$alpha_C_boost * coverage
      effects$theta_i_boost <- effects$theta_i_boost +
        int_effects$theta_i_boost * coverage * (1 + int_effects$compliance_boost * coverage)
      effects$C_intensity <- coverage
    }
  }

  return(effects)
}

simulate_year <- function(initial_state, intervention_type, coverage,
                          region_type, days,
                          params_epi = params_epidemiology,
                          params_behav = params_behavior) {

  m_u <- switch(region_type,
                "dry" = params_epi$m_u_dry,
                "moderate" = params_epi$m_u_mod,
                "wet" = params_epi$m_u_wet,
                params_epi$m_u_mod)

  int_effects <- get_intervention_effects(intervention_type, coverage, m_u, params_epi)
  combined_params <- c(int_effects, params_behav)

```

```r
  y0 <- c(S = as.numeric(initial_state["S"]),
          I = as.numeric(initial_state["I"]),
          R = as.numeric(initial_state["R"]),
          B = as.numeric(initial_state["B"]))

  times <- seq(0, days, by = 1)

  tryCatch({
    solution <- ode(y = y0, times = times, func = combined_odes,
                    parms = combined_params, method = "lsoda")

    final_idx <- nrow(solution)
    final_state <- solution[final_idx, c("S", "I", "R", "B")]

    I_trajectory <- solution[, "I"]
    person_days <- sum(I_trajectory[-1] + I_trajectory[-length(I_trajectory)]) / 2 *
      params_epi$N

    E_vals <- pmin(1, pmax(0,
                           params_behav$theta_B * solution[, "B"] +
                             params_behav$theta_I * solution[, "I"] +
                             combined_params$theta_i_boost -
                             params_behav$theta_C * params_behav$c_effort
    ))

    return(list(
      final_state = final_state,
      person_days = person_days,
      trajectories = as.data.frame(solution),
      effort_trajectory = E_vals,
      success = TRUE,
      intervention = intervention_type,
      coverage = coverage,
      region = region_type
    ))

  }, error = function(e) {
    warning(paste("ODE solver failed for", intervention_type, ":", e$message))
    return(list(
      final_state = initial_state,
      person_days = as.numeric(initial_state["I"]) * params_epi$N * days,
      trajectories = NULL,
      effort_trajectory = NULL,
      success = FALSE,
      intervention = intervention_type,
      coverage = coverage,
      region = region_type
    ))
  })
}

# ============================================================================
# 3. OPTIMIZATION FUNCTIONS - COMPLETE
# ============================================================================

generate_actions <- function() {
  interventions <- c("none", "LLIN", "IRS", "IPT", "ACT", "vaccine", "campaign")
  actions <- list()

  actions[["none_0"]] <- list(type = "none", coverage = 0, cost = 0)

  for (int in setdiff(interventions, "none")) {
    for (cov in params_interventions$coverage_levels) {
      actions[[paste0(int, "_", cov)]] <- list(
        type = int, coverage = cov,
        cost = params_interventions$cost[int] * cov
      )
    }
  }

  key_combinations <- list(
    c("LLIN", "ACT"),
    c("LLIN", "campaign"),
```

```
    c("ACT", "campaign")
  )

  for (combo in key_combinations) {
    combo_name <- paste(combo, collapse = "_")
    for (cov in params_interventions$coverage_levels) {
      cost <- sum(sapply(combo, function(x) params_interventions$cost[x] * cov))
      actions[[paste(combo_name, cov, sep = "_")]] <- list(
        type = combo_name, coverage = cov, cost = cost
      )
    }
  }

  return(actions)
}

precompute_transitions <- function(regions, initial_conditions, actions,
                                   params_epi = params_epidemiology) {
  transitions <- list()
  costs <- list()

  baseline_person_days <- list()
  for (climate in unique(regions$climate)) {
    cat("  Computing", SIM_YEARS, "-year baseline for", climate, "climate...\n")
    res_none <- simulate_year(
      initial_state = initial_conditions[[climate]],
      intervention_type = "none", coverage = 0,
      region_type = climate, days = SIM_DAYS * SIM_YEARS, params_epi = params_epi
    )
    baseline_person_days[[climate]] <- res_none$person_days
  }

  total_combinations <- nrow(regions) * length(actions)
  counter <- 0

  for (i in 1:nrow(regions)) {
    region <- regions[i, ]
    climate <- region$climate
    dist_cost <- region$dist_cost

    cost_multiplier <- switch(dist_cost, "low" = 0.8, "medium" = 1.0, "high" = 1.2)

    for (action_name in names(actions)) {
      counter <- counter + 1
      if (counter %% 10 == 0) {
        cat(sprintf("  Progress: %d/%d (%.1f%%)\r",
                    counter, total_combinations,
                    100 * counter / total_combinations))
      }

      action <- actions[[action_name]]

      result <- simulate_year(
        initial_state = initial_conditions[[climate]],
        intervention_type = action$type,
        coverage = action$coverage,
        region_type = climate, days = SIM_DAYS * SIM_YEARS,
        params_epi = params_epi
      )

      trans_key <- paste(climate, dist_cost, action_name, sep = "|")
      baseline_pd <- baseline_person_days[[climate]]
      avoided_pd <- max(0, baseline_pd - result$person_days)

      transitions[[trans_key]] <- list(
        initial = initial_conditions[[climate]],
        final = result$final_state,
        person_days = result$person_days,
        avoided_person_days = avoided_pd,
        success = result$success
      )

      costs[[trans_key]] <- (action$cost %||% 0) * params_epi$N * cost_multiplier * SIM_
```

```
    YEARS
    }
  }
  cat("\n")

  return(list(transitions = transitions, costs = costs))
}

extract_region <- function(key) {
  parts <- strsplit(key, "\\|")[[1]]
  return(paste(parts[1], parts[2], sep = "|"))
}

solve_ilp_optimization <- function(transitions_data, budget, verbose = TRUE) {
  all_trans <- transitions_data$transitions
  all_costs <- transitions_data$costs
  keys <- names(all_trans)

  objective_fn <- sapply(keys, function(k) all_trans[[k]]$avoided_person_days %||% 0)
  cost_vector <- sapply(keys, function(k) all_costs[[k]] %||% 0)

  valid_indices <- which(objective_fn > 0)
  objective_fn <- objective_fn[valid_indices]
  cost_vector <- cost_vector[valid_indices]
  keys_valid <- keys[valid_indices]

  if (length(keys_valid) == 0) {
    if (verbose) cat("ILP: No valid candidates found (APD <= 0).\n")
    return(list(objective = 0, total_cost = 0, num_selected = 0, solution = list()))
  }

  constraint_matrix <- matrix(cost_vector, nrow = 1)
  constraint_direction <- "<="
  constraint_rhs <- budget

  lp_model <- lpSolve::lp(
    direction = "max",
    objective.in = objective_fn,
    const.mat = constraint_matrix,
    const.dir = constraint_direction,
    const.rhs = constraint_rhs,
    all.bin = TRUE
  )

  if (lp_model$status != 0) {
    if (verbose) cat(sprintf("ILP Solver failed (Status: %d).\n", lp_model$status))
    return(list(objective = 0, total_cost = 0, num_selected = 0, solution = list()))
  }

  solution_vector <- lp_model$solution
  selected_indices <- which(solution_vector == 1)
  selected_keys <- keys_valid[selected_indices]

  selected_cost <- sum(cost_vector[selected_indices])
  selected_apd <- sum(objective_fn[selected_indices])

  selected_regions <- unique(sapply(selected_keys, extract_region))
  total_regions <- length(unique(sapply(keys, extract_region)))

  if (verbose) {
    cat(sprintf("\nINTEGER LINEAR PROGRAM (TRUE OPTIMUM):\n"))
    cat(sprintf("  Actions selected: %d\n", length(selected_keys)))
    cat(sprintf("  Districts covered: %d out of %d\n", length(selected_regions), total_
    regions))
    cat(sprintf("  Total %d-year cost: $%s\n", SIM_YEARS, format(round(selected_cost),
    big.mark = ",")))
    cat(sprintf("  Total %d-year avoided person-days: %s\n", SIM_YEARS, format(round(
    selected_apd), big.mark = ",")))
    cat(sprintf("  Remaining budget: $%s (%.1f%% of total)\n",
                format(round(budget - selected_cost), big.mark = ","),
                ((budget - selected_cost) / budget) * 100))
    pd_per_k <- (selected_apd / max(1, selected_cost)) * 1000
    cat(sprintf("  Average cost-effectiveness: %.1f person-days per $1,000\n", pd_per_k)
```

```
    )
  }

  return(list(
    objective = selected_apd,
    total_cost = selected_cost,
    num_selected = length(selected_keys),
    solution = setNames(as.list(solution_vector[selected_indices]), selected_keys),
    selected_regions = selected_regions
  ))
}

solve_greedy_ce_optimization <- function(transitions_data, budget, verbose = TRUE) {
  all_trans <- transitions_data$transitions
  all_costs <- transitions_data$costs
  keys <- names(all_trans)

  if (length(all_trans) == 0) {
    return(list(solution = NULL, objective = 0, status = "No transitions"))
  }

  ce_ratios <- sapply(keys, function(key) {
    cost <- as.numeric(all_costs[[key]] %||% NA)
    avoided <- as.numeric(all_trans[[key]]$avoided_person_days %||% NA)

    if (!is.na(cost) && !is.na(avoided)) {
      if (cost > 0 && avoided > 0) {
        return(avoided / cost)
      } else if (cost == 0 && avoided > 0) {
        return(Inf)
      }
    }
    return(-Inf)
  }, USE.NAMES = TRUE)

  sorted_keys <- names(sort(ce_ratios[is.finite(ce_ratios)], decreasing = TRUE))

  remaining_budget <- budget
  solution <- list()
  total_avoided <- 0
  total_cost <- 0
  selected_regions <- character(0)
  total_regions <- length(unique(sapply(keys, extract_region)))

  for (key in sorted_keys) {
    cost <- as.numeric(all_costs[[key]])
    avoided <- as.numeric(all_trans[[key]]$avoided_person_days)
    region <- extract_region(key)

    if (!is.na(cost) && !is.na(avoided) &&
        cost <= remaining_budget && avoided > 0 &&
        !(region %in% selected_regions)) {

      solution[[key]] <- 1
      selected_regions <- c(selected_regions, region)
      remaining_budget <- remaining_budget - cost
      total_avoided <- total_avoided + avoided
      total_cost <- total_cost + cost
    }
  }

  if (verbose) {
    cat(sprintf("\nCE-FOCUSED OPTIMIZATION (ONE INTERVENTION PER DISTRICT):\n"))
    cat(sprintf("  Actions selected: %d\n", length(solution)))
    cat(sprintf("  Districts covered: %d out of %d\n", length(selected_regions), total_
    regions))
    cat(sprintf("  Total %d-year cost: $%s\n", SIM_YEARS, format(round(total_cost), big.
    mark = ",")))
    cat(sprintf("  Total %d-year avoided person-days: %s\n", SIM_YEARS,
                format(round(total_avoided), big.mark = ",")))
    cat(sprintf("  Remaining budget: $%s (%.1f%% of total)\n",
                format(round(remaining_budget), big.mark = ","),
                remaining_budget / budget * 100))
```

```
    cat(sprintf("  Average cost-effectiveness: %.1f person-days per $1,000\n",
                (total_avoided / max(1, total_cost)) * 1000))
  }

  return(list(
    solution = solution,
    objective = total_avoided,
    total_cost = total_cost,
    status = "CE-focused solution",
    remaining_budget = remaining_budget,
    num_selected = length(solution),
    selected_regions = selected_regions
  ))
}

run_complete_optimization <- function(budget) {
  cat("\n\
                                                                    n
    \n")
  cat(sprintf("COMPLETE MALARIA INTERVENTION OPTIMIZATION (%d YEARS)\n", SIM_YEARS))
  cat("

    \n\n")

  cat("Configuration:\n")
  cat(sprintf("  Budget (%d Years): $%s\n", SIM_YEARS, format(budget, big.mark = ",")))
  cat(sprintf("  Districts: %d climate x distance combinations\n", nrow(regions)))
  cat(sprintf("  Population: %s per district\n", format(params_epidemiology$N, big.mark
    = ",")))

  cat("\nStep 1: Generating action space...\n")
  actions <- generate_actions()
  cat(sprintf("  Generated %d possible interventions\n", length(actions)))

  cat("\nStep 2: Computing state transitions...\n")
  total_combinations <- nrow(regions) * length(actions)
  cat(sprintf("  Total simulations: %d districts x %d interventions = %d\n",
              nrow(regions), length(actions), total_combinations))
  cat(sprintf("  Each simulation runs for %d years (%d days total).\n", SIM_YEARS, SIM_
    DAYS * SIM_YEARS))

  start_time <- Sys.time()
  transitions_data <- precompute_transitions(regions, initial_conditions, actions)
  end_time <- Sys.time()
  time_taken <- round(as.numeric(difftime(end_time, start_time, units = "mins")), 1)

  cat(sprintf("\ n   Computation complete in %.1f minutes\n", time_taken))

  cat("\nStep 3: Running optimization algorithms...\n")

  cat("\n--- METHOD 1: Integer Linear Program (True Optimum) ---\n")
  result_ilp <- solve_ilp_optimization(transitions_data, budget, verbose = TRUE)

  cat("\n--- METHOD 2: Cost-Effectiveness Focused (Greedy Approximation) ---\n")
  result_ce <- solve_greedy_ce_optimization(transitions_data, budget, verbose = TRUE)

  cat("\nStep 4: Comparing optimization methods...\n")

  ilp_pd_per_k <- (result_ilp$objective / max(1, result_ilp$total_cost)) * 1000
  ce_pd_per_k <- (result_ce$objective / max(1, result_ce$total_cost)) * 1000

  cat(paste(rep("   ", 90), collapse = ""), "\n")
  cat(sprintf("%-25s %-10s %-15s %-20s %-10s %-10s\n",
              "Method", "Actions", "Total Cost", "Avoided PD (5Y)", "Used %", "PD/$1k"))
  cat(paste(rep("   ", 90), collapse = ""), "\n")

  cat(sprintf("%-25s %-10d $%-14s %-20s %-10.1f %-10.0f\n",
              "ILP (True Optimum)",
              result_ilp$num_selected,
              format(round(result_ilp$total_cost), big.mark = ","),
              format(round(result_ilp$objective), big.mark = ","),
              (result_ilp$total_cost / budget) * 100,
              ilp_pd_per_k))
```

```
535
536   cat(sprintf("%-25s %-10d $%-14s %-20s %-10.1f %-10.0f\n",
537               "CE-Focused (Greedy)",
538               result_ce$num_selected,
539               format(round(result_ce$total_cost), big.mark = ","),
540               format(round(result_ce$objective), big.mark = ","),
541               (result_ce$total_cost / budget) * 100,
542               ce_pd_per_k))
543
544   cat(paste(rep("   ", 90), collapse = ""), "\n")
545
546   cost_diff <- result_ilp$total_cost - result_ce$total_cost
547   pd_diff <- result_ilp$objective - result_ce$objective
548
549   cat(sprintf("\nComparison Summary (ILP vs. Greedy):\n"))
550   cat(sprintf("  Additional cost in ILP: $%s\n", format(round(cost_diff), big.mark = ","
        )))
551   cat(sprintf("  Additional PD avoided (%d Years) in ILP: %s (%.1f%% increase)\n",
552               SIM_YEARS,
553               format(round(pd_diff), big.mark = ","),
554               (pd_diff / max(1, result_ce$objective)) * 100))
555
556   cat("\
                                                                          n
        \n")
557   cat(sprintf("%d-YEAR OPTIMIZATION COMPLETE\n", SIM_YEARS))
558   cat("
        \n\n")
559
560   return(list(
561     actions = actions,
562     transitions_data = transitions_data,
563     ilp_optimum = result_ilp,
564     ce_focused = result_ce,
565     timestamp = Sys.time()
566   ))
567 }
568
569 # ================================================================================
570 # ANALYSIS FUNCTIONS
571 # ================================================================================
572
573 analyze_campaign_impact <- function(results) {
574   cat("\n")
575   cat("
        \n")
576   cat("BEHAVIORAL MODEL & CAMPAIGN IMPACT ANALYSIS\n")
577   cat("
        \n\n")
578
579   # Extract ILP solution
580   ilp_keys <- names(results$ilp_optimum$solution)
581
582   # Categorize by intervention type
583   has_campaign <- grepl("campaign", ilp_keys)
584   campaign_interventions <- ilp_keys[has_campaign]
585   non_campaign <- ilp_keys[!has_campaign]
586
587   cat(sprintf("Total interventions selected: %d\n", length(ilp_keys)))
588   cat(sprintf("  With campaigns: %d (%.1f%%)\n",
589               length(campaign_interventions),
590               100 * length(campaign_interventions) / length(ilp_keys)))
591   cat(sprintf("  Without campaigns: %d (%.1f%%)\n\n",
592               length(non_campaign),
593               100 * length(non_campaign) / length(ilp_keys)))
594
595   # Extract campaign contribution to total impact
596   campaign_apd <- sum(sapply(campaign_interventions, function(k) {
597     results$transitions_data$transitions[[k]]$avoided_person_days %||% 0
598   }))
```

```r

  total_apd <- results$ilp_optimum$objective

  cat("Campaign Contribution:\n")
  cat(sprintf("  Direct person-days avoided: %s\n",
              format(round(campaign_apd), big.mark = ",")))
  cat(sprintf("  Percentage of total impact: %.1f%%\n\n",
              100 * campaign_apd / total_apd))

  # Cost analysis
  campaign_cost <- sum(sapply(campaign_interventions, function(k) {
    results$transitions_data$costs[[k]] %||% 0
  }))

  total_cost <- results$ilp_optimum$total_cost

  cat("Campaign Investment:\n")
  cat(sprintf("  Total campaign costs: $%s\n",
              format(round(campaign_cost), big.mark = ",")))
  cat(sprintf("  Percentage of total budget: %.1f%%\n",
              100 * campaign_cost / total_cost))
  cat(sprintf("  Campaign cost-effectiveness: %.0f PD/$1k\n\n",
              (campaign_apd / campaign_cost) * 1000))

  # Show top campaign interventions
  campaign_data <- data.frame(
    key = campaign_interventions,
    apd = sapply(campaign_interventions, function(k) {
      results$transitions_data$transitions[[k]]$avoided_person_days
    }),
    cost = sapply(campaign_interventions, function(k) {
      results$transitions_data$costs[[k]]
    }),
    stringsAsFactors = FALSE
  )

  if (nrow(campaign_data) > 0) {
    campaign_data$ce_ratio <- campaign_data$apd / campaign_data$cost * 1000
    campaign_data <- campaign_data[order(-campaign_data$apd), ]

    cat("Top Campaign Interventions:\n")
    cat(paste(rep("   ", 70), collapse = ""), "\n")
    n_show <- min(10, nrow(campaign_data))
    for (i in 1:n_show) {
      row <- campaign_data[i, ]
      cat(sprintf("%-40s %12s %10.0f PD/$1k\n",
                  row$key,
                  format(round(row$apd), big.mark = ","),
                  row$ce_ratio))
    }
  }

  cat("\n")
  invisible(campaign_data)
}

# ============================================================================
# MAIN EXECUTION
# ============================================================================

BUDGET_TO_RUN <- 2300000
cat("\nStarting complete 5-year optimization...\n")
complete_results <- run_complete_optimization(budget = BUDGET_TO_RUN)

# Analyze campaign impact
analyze_campaign_impact(complete_results)

cat("\n\nModel execution complete!\n")
cat("Results saved in 'complete_results' variable.\n")
cat("\nTo save: saveRDS(complete_results, 'malaria_optimization_5year.rds')\n")
cat("To analyze campaigns: analyze_campaign_impact(complete_results)\n\n")
```

Listing 1: Final R implementation (full).

References

[1] Fenichel, E. P., Castillo-Chavez, C., Ceddia, M. G., Chowell, G., Parra, P. A. G., Hickling, G. J., ..., & Villalobos, C. (2011). Adaptive human behavior in epidemiological models. *Proceedings of the National Academy of Sciences*, **108**(15), 6306–6311.

[2] Dudley, H. J., Goenka, A., Orellana, C. J., & Martonosi, S. E. (2016). Multi-year optimization of malaria intervention: a mathematical model. *Malaria journal*, **15**(1), 133.