Risk-Based Group Testing Optimization

Camellini Francesco
Guermazi Youssef
Parietti Alessandro
Ung Seng Iat
Xie Wang Chao

November 2025

Reference

http://pubsonline.informs.org/journal/mnsc

MANAGEMENT SCIENCE
Vol. 65, No. 9, September 2019, pp. 4365–4384
ISSN 0025-1909 (print), ISSN 1526-5501 (online)

Optimal Risk-Based Group Testing

Hrayer Aprahamian,[a] Douglas R. Bish,[b] Ebru K. Bish[b]

[a] Industrial and Systems Engineering, Texas A&M University, College Station, Texas 77843; [b] Grado Department of Industrial and Systems Engineering, Virginia Tech, Blacksburg, Virginia 24061

Contact: hrayer@tamu.edu, http://orcid.org/0000-0002-8750-2366 (HA); drb1@vt.edu (DRB); ebru@vt.edu (EKB)

Received: April 25, 2017
Revised: March 4, 2018; May 3, 2018
Accepted: May 24, 2018
Published Online in Articles in Advance: May 8, 2019

https://doi.org/10.1287/mnsc.2018.3138

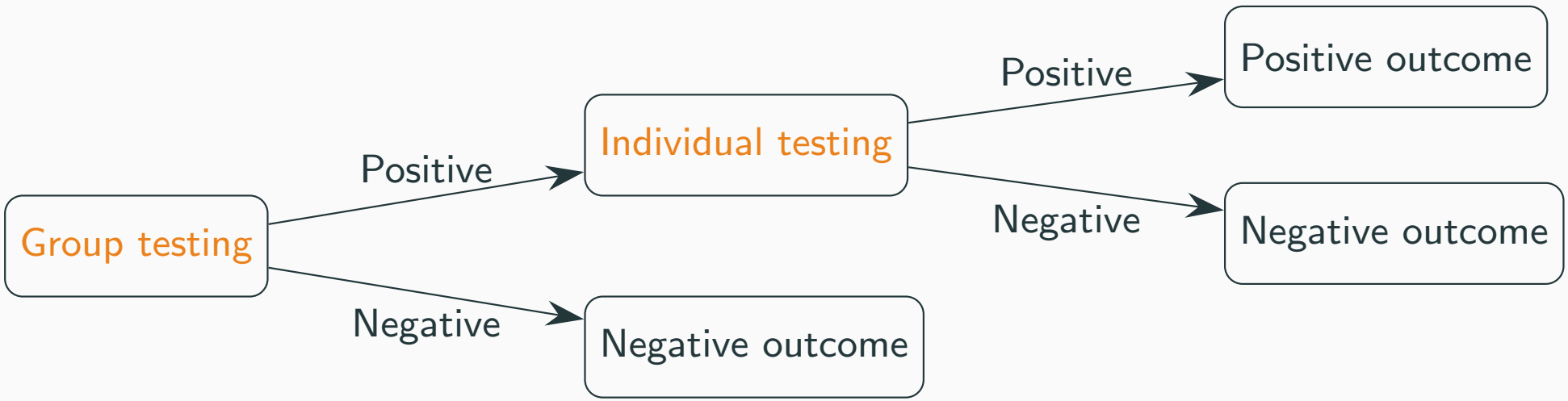
Abstract. Group testing (i.e., testing multiple subjects simultaneously with a single test) is essential for classifying a large population of subjects as *positive* or *negative* for a binary characteristic (e.g., presence of a disease). We study optimal group testing designs under subject-specific risk characteristics and imperfect tests, considering classification accuracy-, efficiency- and equity-based objectives, and characterize important structural properties of optimal testing designs. These properties allow us to model the testing design problems as partitioning problems, develop efficient algorithms, and derive insights on equity versus accuracy trade-off. One of our models reduces to a constrained shortest path problem, for a special case of which we develop a polynomial-time algorithm. We also show that determining an optimal risk-based Dorfman testing scheme that minimizes the expected number of tests is tractable, resolving an open conjecture. We demonstrate the value of optimal risk-based testing schemes with a case study of public health screening.

History: Accepted by Yinyu Ye, optimization.
Funding: This material is based on work supported in part by the National Science Foundation [Grant 1055360]. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.
Supplemental Material: The online appendix is available at https://doi.org/10.1287/mnsc.2018.3138.

Keywords: group testing • Dorfman testing • risk-based testing • classification errors • equity • combinatorial optimization • constrained shortest path

Introduction

Dorfman Testing Scheme



Usage of the model and traditional approach

Group testing is an effective approach for screening large populations for binary outcomes such as diseases, defects, or errors. Especially, it can reduce costs when there are constrained.

Traditional group testing methods use strong assumptions, which we will prove not to be necessary:

- Perfect tests.
- Infinite testing population.
- Homogeneous population.

In our formulation, tests are not perfect, population is finite and every individual has its own risk of positivity.

Notation

In this presentation we are going to use the following notation:

- $\mathbf{S} = (\mathbf{1}, \ldots, \mathbf{N})$: ordered list of subject, with increasing positivity probability.
- $\mathbf{p} = (\mathbf{p^1}, \ldots, \mathbf{p^n})$: risk vector corresponding to S.
- **Se**: sensitivity (true-positive probability).
- **Sp**: sensibility (true-negative probability).
- $\mathbf{\Omega} = \{\mathbf{\Omega_i}\}_{\mathbf{i}=\mathbf{1},\ldots,\mathbf{g}}$: ordered partition of S, with $n_i = |\Omega_i|$.
- $\mathbf{\Omega^I}$: subset of S which is tested individually.
- $\mathbf{\Omega^G}$: subset of S which is tested as a group.
- $\mathbf{I^m}(\mathbf{\Omega})$, $\mathbf{FN^m}(\mathbf{\Omega})$, $\mathbf{FP^m}(\mathbf{\Omega})$: positivity, false-negativity and false-positivity status of subject m, given partition Ω.
- $\mathbf{T}(\mathbf{\Omega})$, $\mathbf{FN}(\mathbf{\Omega})$, $\mathbf{FP}(\mathbf{\Omega})$: total number of tests, false-negatives and false-positives in group Ω.

T and FN probability

$$\mathbb{E}[T(\Omega_i)] = \begin{cases} 1, & \text{if } n_i = 1, \\ 1 + n_i \underbrace{\left(Se - (Se + Sp - 1) \prod_{m \in \Omega_i} (1 - p^m) \right)}_{\text{mean probability of positive group test}}, & \text{otherwise.} \end{cases}$$

$$\mathbb{E}[FN^m] = \begin{cases} (1 - Se)p^m, & \text{if } m \in \Omega^I, \\ (\underbrace{Se(1 - Se)}_{\text{TP group, FN individual}} + \underbrace{(1 - Se)}_{\text{FN group}})p^m = (1 - Se^2)p^m, & \text{if } m \in \Omega^G. \end{cases}$$

Implementation: $E[T]$ and $E[FN]$

```python
def e_fn(group_risks: np.ndarray, Se: float) -> float:

    n = group_risks.size
    if n == 0: return 0.0

    # Individual testing: single miss probability
    if n == 1:
        return (1.0 - Se) * group_risks[0]

    # Group testing: higher miss probability (dilution/two-stage)
    return (1.0 - Se**2) * float(group_risks.sum())

def e_t(group_risks: np.ndarray, Se: float, Sp: float) -> float:

    n = group_risks.size
    if n == 0: return 0.0
    if n == 1: return 1.0

    # Probability that ALL are healthy (observed as negative)
    prod_1mp = float(np.prod(1.0 - group_risks))
    C = Se + Sp - 1.0

    # Cost = 1 + n * (Prob Group Positive)
    return 1.0 + n * (Se - C * prod_1mp)
```

FP probability

The probability of a FP can be easily calculated in case of $m \in \Omega^I$ as

$$\mathbb{E}[FP^m] = (1 - Sp)(1 - p^m).$$

When $m \in \Omega^G$, the probability can be written as

$$\mathbb{E}[FP^m] = \underbrace{(1 - Sp)^2 \prod_{k \in \Omega_i \setminus \{m\}} (1 - p^k)}_{\text{FP group, FP individual}} +$$

$$+ \underbrace{Se(1 - Sp)\left(1 - \prod_{k \in \Omega_i \setminus \{m\}} (1 - p^k)\right)}_{\text{TP group, FP individual}} (1 - p^m)$$

$$= (1 - Sp)Se(1 - p^m) - (1 - Sp)(Se + Sp - 1) \prod_{k \in \Omega_i} (1 - p^k)$$

Implementation: Expected False Positives ($E[FP]$)

```
def e_fp(group_risks: np.ndarray, Se: float, Sp: float) -> float:
    """
    Expected false positives. Depends on Specificity (Sp).
    """
    n = group_risks.size
    if n == 0: return 0.0

    sum_1mp = float((1.0 - group_risks).sum())
    if n == 1:
        return (1.0 - Sp) * sum_1mp

    # For groups, FP occurs if Group is Positive AND Indiv is
     Positive
    prod_1mp = float(np.prod(1.0 - group_risks))
    C = Se + Sp - 1.0

    term1 = (1.0 - Sp) * Se * sum_1mp
    term2 = n * (1.0 - Sp) * C * prod_1mp
    return term1 - term2
```

Data: Real World

We are going to implement the optimization on the following dataset, which represents the risk for chlamydia in different risk groups:

Gender	Race/ ethnicity	Age group (years)	Risk (prevalence) (%)	Proportion in general population (%)
Female	Hispanic	15–24	6.54	1.41
		Other	0.65	7.01
	Black	15–24	19.19	1.07
		Other	1.22	5.67
	Other	15–24	4.38	4.29
		Other	0.25	31.31
Male	Hispanic	15–24	1.78	1.53
		Other	0.36	7.16
	Black	15–24	7.45	1.09
		Other	1.05	5.08
	Other	15–24	1.20	4.51
		Other	0.17	29.87

Figure 1: Table 1

Each time we use the data, we run 100 Monte Carlo simulations on populations of 100 individuals generated from the given distribution.

We set $Sp = 0.95$ and $Se = 0.95$.

Data: Artificial

We also implement the optimization on a randomized dataset, which is produced by the following code:

```
while True:
    risks = np.random.normal(loc=3, scale=2, size=12)
    if np.all(risks > 0):
        break

populations = np.random.rand(12)
populations = populations / populations.sum() * 100

table2_data = list(zip(np.round(risks, 2), np.round(populations,2)))
```

We use the data in the same way as before, running Monte Carlo simulations with the same values of Sp and Se.

System-Optimal Model (SM)

Objective Function

The objective is to minimize a weighted sum of number of tests and classification errors *FN* and *FP*.

$$\underset{\Omega}{\text{minimize}} \quad \lambda_1 \mathbb{E}[FN(\Omega)] + \lambda_2 \mathbb{E}[FP(\Omega)] + (1 - \lambda_1 - \lambda_2)\mathbb{E}[T(\Omega)]$$

With $\lambda_1, \lambda_2 \in [0, 1]$ and $\lambda_1, \lambda_2 \leq 1$.

In our implementation we set:

- Cost of testing: \$55
- Cost of FP: \$55 (cost of confirmatory test)
- Cost of FN: \$2927

Which leads to $\lambda_1 = 0.96$ and $\lambda_2 = 0.02$

Properties

The most efficient way to solve this problem is to represent it as a shortest path problem. To do so, we rely on the following properties:

- There exists an optimal partition that is an ordered partition of S.
- The subjects in Ω^I correspond to the highest-risk individuals in set S.
- If $\lambda_1 + \lambda_2 = 1$, no group can contain more than three subjects.
- If $\lambda_1 = 1$, it is optimal to test everyone individually.
- If $\lambda_2 = 1$, it is optimal to test at most one person individually.

The non-increasing size of the partitions, which might seem intuitive, is guaranteed only if $\lambda_1 + \lambda_2 = 1$ and $p^N \leq 1/3$. Therefore, it is not a general property.

Graph definition

We define an acyclic directed graph $G = (V, E)$.

- $V = (1, 2, ..., N + 1)$ is the vertex set.
- $E = \{(i, j) \in V : i < j\}$ is the edge set.

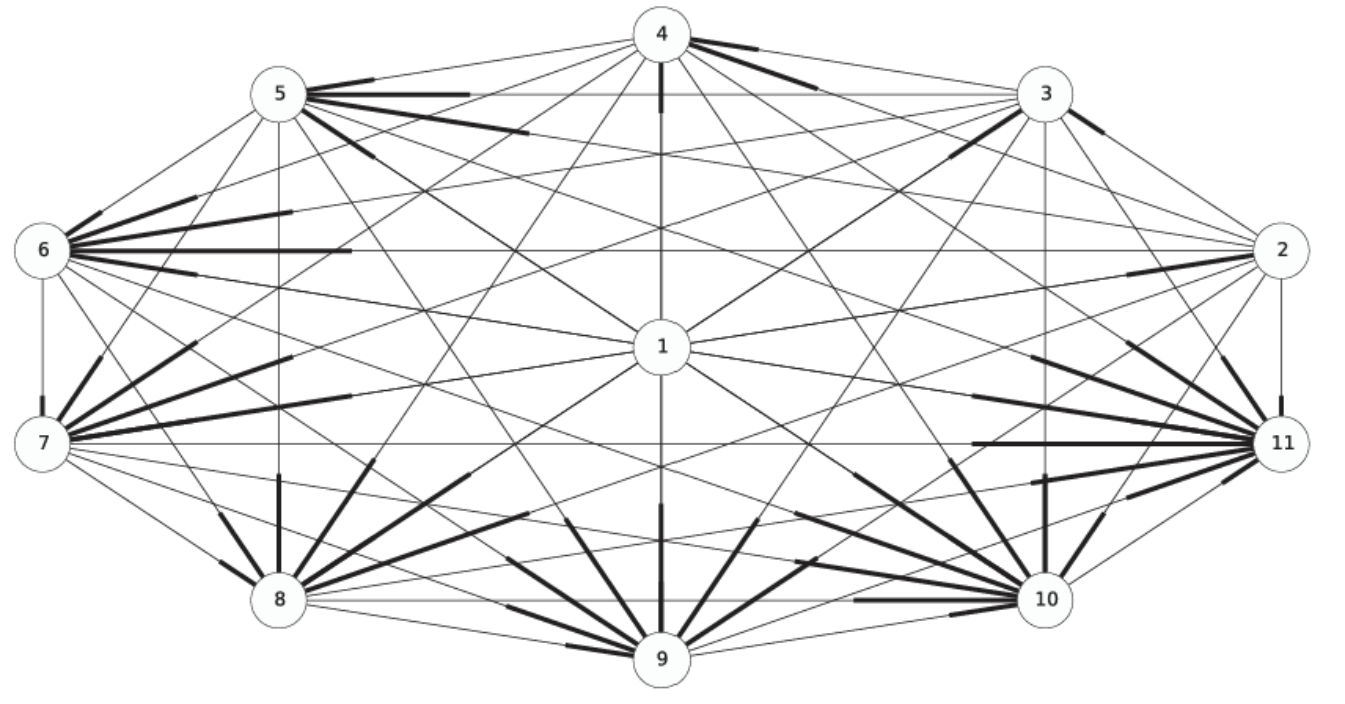
Every edge represents the group $\Omega_{ij} = (i, , i + 1..., j - 1)$. For example:

$$S = \underbrace{(1, 2, 3, 4)}_{(1,5)} \cup \underbrace{(5, 6, 7)}_{(5,8)} \cup \underbrace{(8)}_{(8,9)}$$

We define the weight c_{ij} corresponding to edge (i, j) as:

$$c_{ij} = \lambda_1 \mathbb{E}[FN(\Omega_{ij})] + \lambda_2 \mathbb{E}[FP(\Omega_{ij})] + (1 - \lambda_1 - \lambda_2)\mathbb{E}[T(\Omega_{ij})]$$

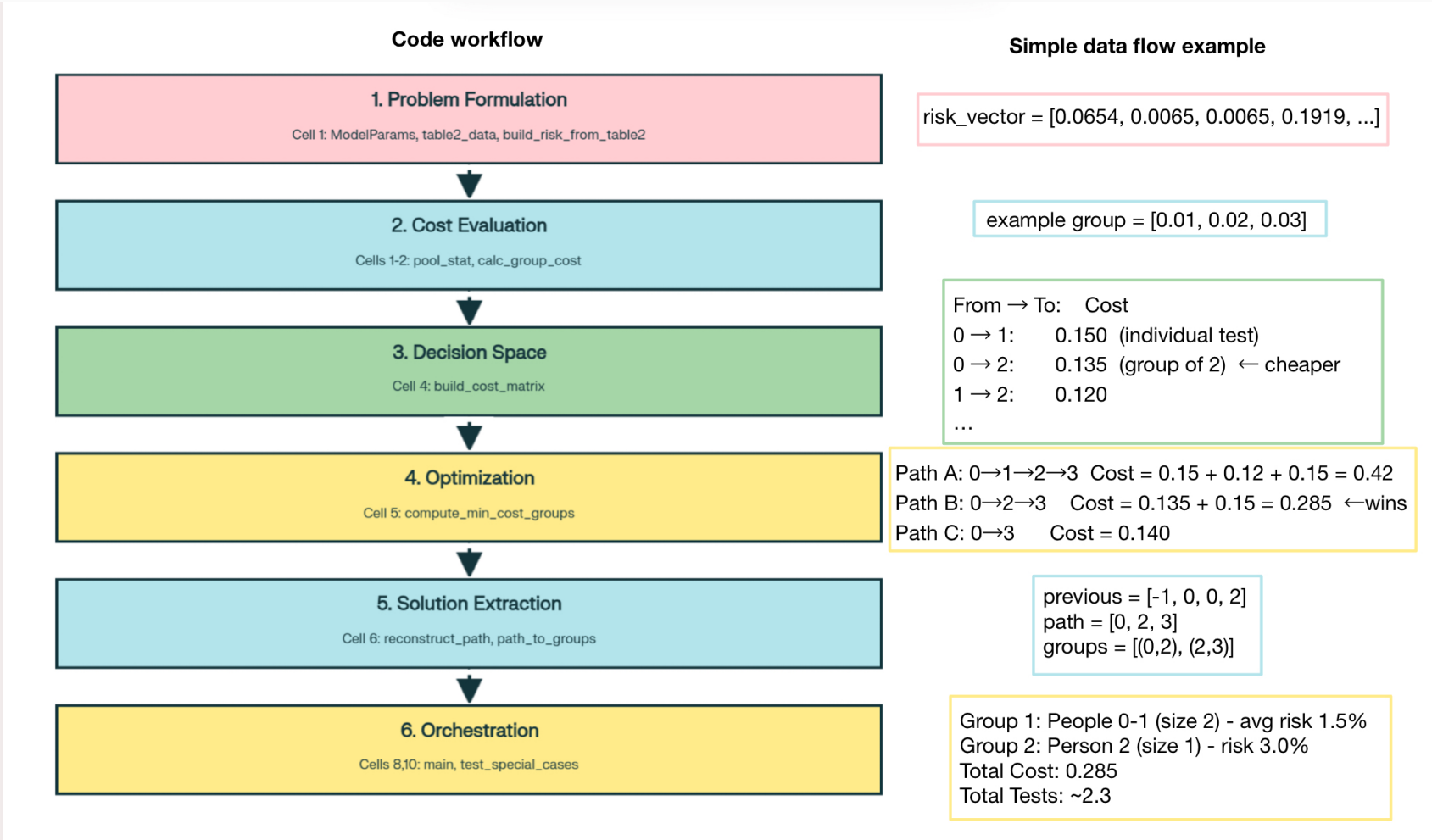
Shortest path problem and complexity



Finding the shortest path between 1 and $N+1$ is equivalent to finding the optimal solution to our optimization problem.

The computational complexity of solving this problem is $\mathcal{O}(N^2)$.

Implementation



Implementation: Group Cost Calculation

First, we translate the objective function into a Python function. This calculates the weight c_{ij} for a specific edge (group).

$$Cost = \lambda_1 \mathbb{E}[FN] + \lambda_2 \mathbb{E}[FP] + (1 - \lambda_1 - \lambda_2)\mathbb{E}[T]$$

```
1  def calc_group_cost(risk_group, params: ModelParams):
2      # Calculate expectations for the specific group
3      expected_tests, expected_fn, expected_fp = pool_stat(risk_group,
         params)
4
5      # Apply the objective function weights
6      cost = (params.lambda_1 * expected_fn +
7              params.lambda_2 * expected_fp +
8              (1 - params.lambda_1 - params.lambda_2) * params.
       test_cost * expected_tests)
9
10     return cost
```

Implementation: Graph Construction

We build the Cost Matrix representing the acyclic directed graph.

- **Nodes**: 0 to N (representing cut points).
- **Edges**: From i to j representing a group of subjects $[i, j)$.

```python
def build_cost_matrix(risk_vector, params: ModelParams):
    n = len(risk_vector)
    # Initialize with infinity (no edge)
    cost_matrix = np.full((n+1, n+1), np.inf)

    for i in range(n):
        # Create edges to all valid next nodes j
        # Limit group size for performance if needed
        limit = min(n+1, i + params.max_group_size + 1)

        for j in range(i+1, limit):
            group_risks = risk_vector[i:j]
            # Assign weight c_ij to edge (i, j)
            cost_matrix[i, j] = calc_group_cost(group_risks, params)

    cost_matrix[n, n] = 0 # End node cost
    return cost_matrix
```

Implementation: Shortest Path (Dijkstra)

Finally, we solve the problem using Dijkstra's algorithm to find the optimal partition Ω^*.

```
def compute_min_cost_groups(cost_matrix, start_node):
    n = len(cost_matrix)
    total_cost = np.full(n, np.inf)
    total_cost[start_node] = 0
    previous = np.full(n, -1, dtype=int)

    # ... (Standard Dijkstra Loop logic omitted for brevity) ...
    # inside the loop:

        # If a cheaper path is found:
        new_total_cost = total_cost[current_node] + cost_matrix[
     current_node][neighbor]

        if new_total_cost < total_cost[neighbor]:
            total_cost[neighbor] = new_total_cost
            previous[neighbor] = current_node

    return total_cost, previous
```

SM Results: Real-world dataset

We ran the System-Optimal Model for $\lambda_1 = 0.96, \lambda_2 = 0.02$.

Results (Mean over 100 Simulations):

Metric	Mean Value	Insight
Total Cost	0.4773	–
Exp. Tests ($E[T]$)	15.37	$\sim$85% reduction vs Individual
Exp. False Neg. ($E[FN]$)	0.1653	High Accuracy ($\lambda_1 = 0.96$)
Exp. False Pos. ($E[FP]$)	0.5573	Acceptable trade-off
Avg. Groups	8.03	–

The model prioritizes minimizing False Negatives (weight 0.96) while still achieving a massive reduction in the number of tests required.

SM Results: Artificial dataset

We ran the System-Optimal Model for $\lambda_1 = 0.96, \lambda_2 = 0.02$ on the randomly generated dataset.

Results (Mean over 100 Simulations):

Metric	Mean Value	Insight
Total Cost	1.2054	–
Exp. Tests ($E[T]$)	33.52	∼66% reduction vs Individual
Exp. False Neg. ($E[FN]$)	0.5375	Impact of higher avg risk
Exp. False Pos. ($E[FP]$)	0.9517	–
Avg. Groups	18.50	More fragmentation needed

Due to the higher overall risk in the artificial population, the algorithm creates more groups and requires more tests compared to the real-world case to maintain accuracy.

SM Results: Optimal Partition Structure

We analyze the partition of the last simulation from the real-world dataset. The group sizes adapt dynamically to the risk profile:

- **Low Risk (0.17% – 0.31%):** Subjects are placed in large groups (Size 19 – 26) to maximize efficiency.
- **Medium Risk (0.70% – 1.25%):** Subjects are placed in medium-sized groups (Size 9 – 12).
- **High Risk (4.38% – 19.19%):** Subjects are placed in small groups (Size 5) or tested individually (Size 1) to prevent dilution and minimize False Negatives.

The Shortest Path algorithm automatically identifies that the highest risk individuals (19.19%) must be tested individually, without manual thresholding.

Budget-Constrained Model (BM)

Objective Function

The objective is to minimize a weighted sum of both types of classification errors FN and FP under a testing budget constraint (B).

$$\begin{aligned} \underset{\Omega}{\text{minimize}} \quad & \lambda \mathbb{E}[FN(\Omega)] + (1-\lambda)\mathbb{E}[FP(\Omega)] \\ \text{subject to} \quad & \mathbb{E}[T(\Omega)] \leq B \end{aligned}$$

With $\lambda \in [0, 1]$

Case $\lambda = 1$

If $\lambda = 1$, we are minimizing only $\mathbb{E}[FN]$.

We can build a polynomial time algorithm using the following properties:

- There exists an optimal partition that is an ordered partition of S.
- The subjects in Ω^I correspond to the highest-risk subjects in set S.
- Testing any subject individually reduces $\mathbb{E}[FN]$
- The internal partition of Ω^G doesn't influence $\mathbb{E}[FN]$

Those properties rely on the assumption that testing responses are conditionally independent, given I_i.

If there are multiple optimal solutions, the algorithm delivers the one with the minimum $\mathbb{E}[T]$.

Case $\lambda = 1$

The following algorithm solves the problem for N subjects in $\mathcal{O}(N^3)$.

Algorithm

0. If $B \geq N$, stop. The solution is individual testing.

1. Let $\hat{N} = 2$, $S_1 = \{1, 2\}$ and $S_2 = S - S_2$.

2. Solve SM with $S = S_1$, minimizing only $\mathbb{E}[T]$ ($\lambda_1 = \lambda_2 = 0$). Denote Ω^* the optimal partition of S_1.

3. If $T(\Omega^*) + |S2| \leq B$, the optimal solution is to test S_2 individually and S_1 according to Ω^*.

4. If $\hat{N} = N$ and $\Omega^* > B$, stop; the problem is infeasible.

5. Set $\hat{N} = \hat{N} + 1$, $S_1 = S_1 \cup \{\hat{N}\}$ and $S_2 = S \setminus S_1$. Go to step 2.

Implementation: BM Solver - Setup & Checks

The ‘test‘ function first handles edge cases (infinite budget) and checks basic feasibility against the minimum possible tests.

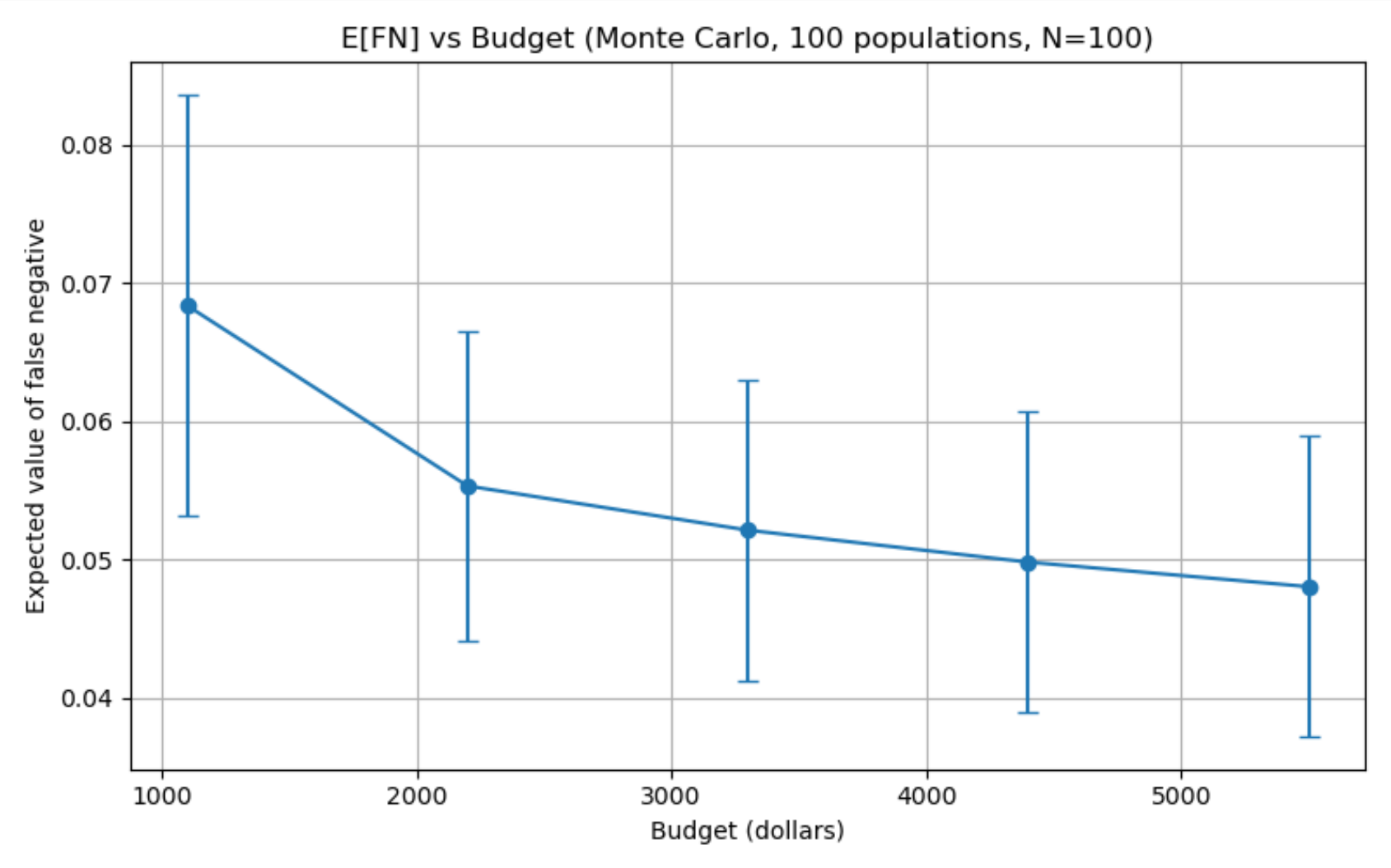
```
def test(risk_vector: np.ndarray, params: ModelParams, B: float) -> Dict:
    Se, Sp = params.sensitivity, params.specificity
    N = risk_vector.size

    # 1. Check if Budget covers Individual Testing (Trivial case)
    if B >= N - 1e-9:
        total_EFN = (1-Se) * float(risk_vector.sum())
        return {"groups": [(i,i+1) for i in range(N)], "ET": N, "EFN": total_EFN}

    # 2. Check Feasibility (Budget < Min Possible Tests?)
    base = sm_minET_fn(risk_vector, params.max_group_size, Se, Sp)
    if B + 1e-9 < base["ET"]:
        return {"error": "INFEASIBLE: Budget too low"}

    # ... (Optimization loop continues in next slide) ...
```

Implementation: BM Solver - Optimization Loop

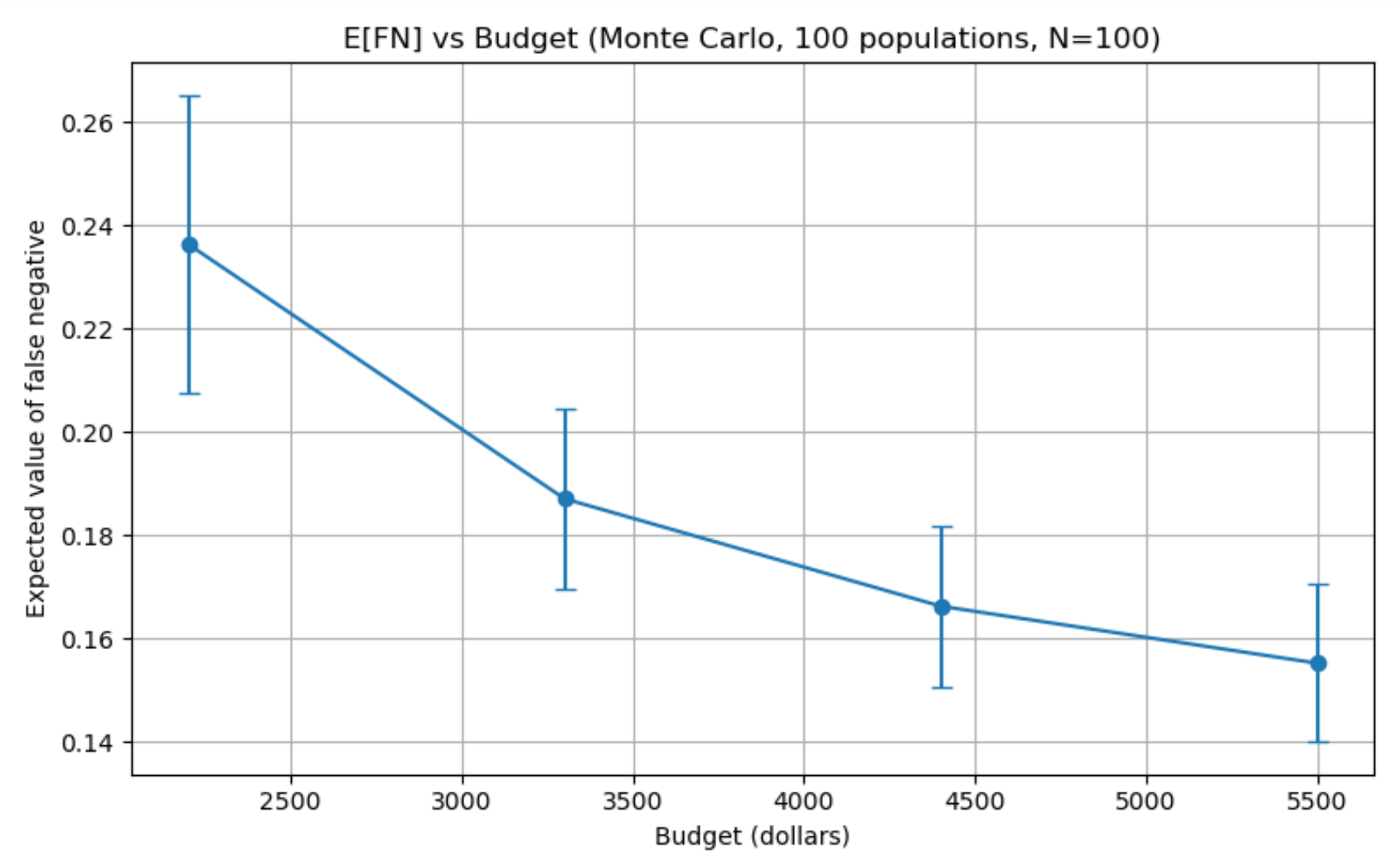
The core loop iterates k to finding the optimal split: minimize $E[T]$ for the low-risk prefix and test the high-risk suffix individually.

```
    # 3. Find optimal split k
    k = 2
    while k <= N:
        # Optimize Prefix (0..k) for Efficiency
        risk_group = risk_vector[:k]
        out = sm_minET_fn(risk_group, params.max_group_size, Se, Sp)
        Z = float(out["ET"])
        single_tests = N - k

        # Check if mixed strategy fits Budget
        if Z + single_tests <= B + 1e-9:
            groups_full = out["groups"] + [(i,i+1) for i in range(k, N)]
            # Calculate Total False Negatives
            EFN = sum(e_fn(risk_vector[i:j], Se) for i,j in groups_full)
            return {"groups": groups_full, "ET": Z+single_tests, "EFN": EFN}
        k += 1

    return {"error": "INFEASIBLE"}
```

Case $\lambda = 1$: Results with Real-World Data



Case $\lambda = 1$: Results with Artificial Data



Case $\lambda = 1$, costly false positives

We now consider a variation of the original problem, where every false positive has a cost. The constraint becomes:

$$\mathbb{E}[T(\Omega)] + \gamma \mathbb{E}[FP(\Omega)] \leq B$$

Algorithm

0. If $B \geq N$, stop. The solution is individual testing.

1. Let $\hat{N} = 2$, $S_1 = \{1, 2\}$ and $S_2 = S - S_2$.

2. Solve SM with $S = S_1$, minimizing both $\mathbb{E}[T]$ and $\mathbb{E}[FP]$ ($\lambda_1 = 0$, $\lambda_2 = \gamma/(1+\gamma)$). Denote Ω^* the optimal partition of S_1.

3. If $T(\Omega^*) + |S2| + \gamma \mathbb{E}[FP] \leq B$, the optimal solution is to test S_2 individually and S_1 according to Ω^*.

4. If $\hat{N} = N$ and $\Omega^* > B$, stop; the problem is infeasible.

5. Set $\hat{N} = \hat{N} + 1$, $S_1 = S_1 \cup \{\hat{N}\}$ and $S_2 = S \setminus S_1$. Go to step 2.

Case $\lambda < 1$

If $\lambda < 1$, we need to formulate the problem in a more general way, for instance as a constrained shortest path problem. This leads to the following integer programming problem, which can be solved with the branch and bound approach.

We define the binary variable x_{ij}, with $1 \leq i < j \leq N + 1$. It is:

- 1 if there exists a subset $\Omega_{ij} = i, ..., j - 1$.
- 0 otherwise.

Case $\lambda < 1$

The objective function is:

$$\underset{x}{\text{minimize}} \quad \sum_{i=1}^{N} \sum_{j=i+1}^{N+1} \left(\lambda \, \mathbb{E}[FN(\Omega_{ij})] + (1-\lambda) \, \mathbb{E}[FP(\Omega_{ij})]\right) x_{ij}$$

The constraints are:

- Every subject needs to be exactly in one subset.

$$\sum_{j=i+1}^{N+1} x_{ij} - \sum_{j=1}^{i-1} x_{ji} = \begin{cases} 1, & \text{if } i = 1, \\ -1, & \text{if } i = N+1, \\ 0, & \text{otherwise,} \end{cases} \qquad \forall i$$

Case $\lambda < 1$

- From every subject starts at most one group (tightening constraint):

$$\sum_{j=i+1}^{N+1} x_{ij} \leq 1 \qquad \forall i$$

- If subject i is tested individually, every following subject is tested individually:

$$x_{j,j+1} \geq x_{i,i+1}, \qquad \forall (i,j) : j > i$$

- The price constraint must be respected:

$$\sum_{i=1}^{N} \sum_{j=i+1}^{N+1} \mathbb{E}[T(\Omega_{ij})]\, x_{ij} + \gamma \mathbb{E}[FP(\Omega_{ij})] x_{ij} \leq B$$

- The variable we have defined are binary:

$$x_{ij} \in \{0,1\} \qquad \forall (i,j)$$

Equity-Based Model

Equity problem

The solution to the SM problem maximizes classification accuracy, but it may be unfair to certain individuals. In particular, some demographic groups may be misclassified more frequently than others under this solution.

For this purpose, we define the inequality aversion parameter α:

- $\alpha \geq 0$.
- The larger α is, the more equitable the solution becomes.
- If $\alpha = 0$, no equity is required.

Optimization formulation

We keep the same formulation as before, but the $(1-\alpha)$ exponent makes improvements for poorly tested individuals more valuable than improvements for well-tested ones, thereby boosting equity.

$$\begin{aligned} \max_{\Omega} \quad & \frac{1}{1-\alpha} \sum_{m \in S} (1 - \lambda \mathbb{E}[FN^m(\Omega)] - (1-\lambda)\mathbb{E}[FP^m(\Omega)])^{1-\alpha} \\ \text{s.t.} \quad & \mathbb{E}[T(\Omega)] \leq B. \end{aligned}$$

We can define the price of fairness as

$$PoF(\alpha) = \frac{f(\alpha) - f(0)}{f(0)}, \qquad f(\alpha) = \lambda \mathbb{E}[FN(\Omega^*(\alpha))] - (1-\lambda)\mathbb{E}[FP(\Omega^*(\alpha))]$$

Case $\alpha \to \infty$, $\lambda = 1$

We consider the problem where we want to minimize overall disparities ($\alpha \to \infty$) in false-negative rates ($\lambda = 1$). In this case the problem can be rewritten as:

$$\begin{aligned} \min_{\Omega} \quad & \max_{m} \{\mathbb{E}[FN^m(\Omega)]\} \\ \text{s.t.} \quad & \mathbb{E}[T(\Omega)] \leq B. \end{aligned}$$

We can linearize the problem by introducing the auxiliary variable z:

$$\begin{aligned} \min_{\Omega,\ z} \quad & z \\ \text{s.t.} \quad & \mathbb{E}[FN^m(\Omega)] \leq z \qquad \forall m \in S \\ & \mathbb{E}[T(\Omega)] \leq B. \end{aligned}$$

Case $\alpha \to \infty$, $\lambda = 1$

The previous problem is still difficult to solve without the following properties, which stand for $\lambda = 1$:

- The internal grouping of Ω^G does not influence the objective value.
- Testing any subject independently minimizes the objective function value.

$$(1 - Se)p^m \leq (1 - Se^2)p^m$$

- Putting the highest risk subjects in Ω^I minimizes the objective function value. In fact, if $p^m \leq p^n$

$$\max\{(1-Se^2)p^m, (1-Se)p^m, (1-Se^2)p^n, (1-Se)p^n\} = (1-Se^2)p^n.$$

Case $\alpha \to \infty$, $\lambda = 1$

The properties we have just presented simplify the problem even further. In fact, we can obtain the optimal solution simply by assigning as many high-risk individuals as possible to Ω^I, subject to the budget constraint.

The algorithm we found for BM under the assumption of $\lambda = 1$ solves this problem.

Case $\alpha \geq 0$, $\lambda = 1$

The previous result is also a consequence of the following theorem:

Theorem

For the equity problem with $\lambda = 1$, there exists an optimal ordered partition that is independent of α.

Therefore, the solution of the problem as $\alpha \to \infty$ coincides with the solution obtained for $\alpha = 0$, which corresponds exactly to the BM formulation.

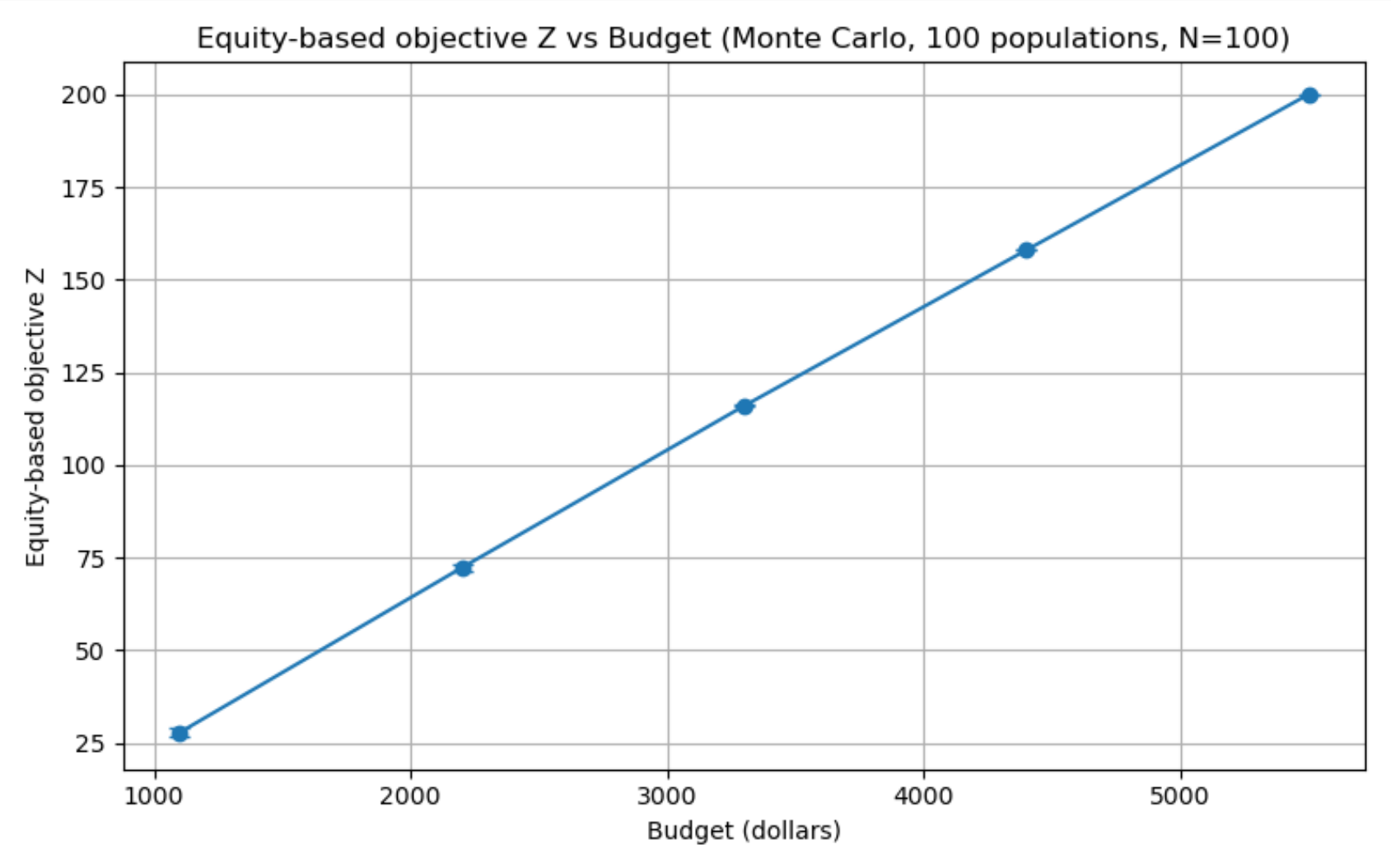
Clearly, if $\lambda = 1$, $PoF(\alpha) = 0$ for any α.

Case $\lambda = 1$: Implementation

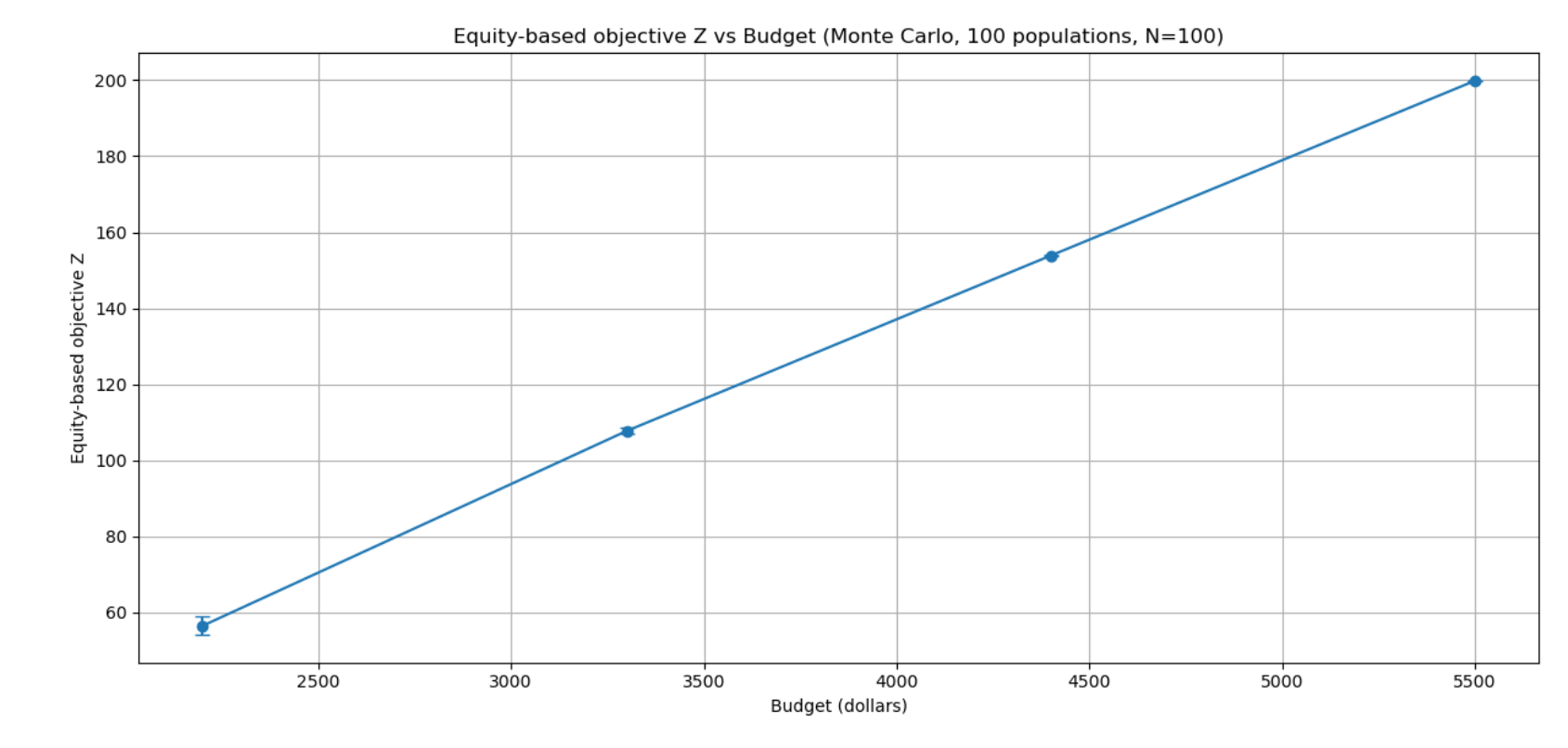
We evaluate the fairness of the solution using the α-fairness metric, with $\alpha = 0.5$. A higher objective value indicates a more equitable solution.

```
def compute_Z(risks: np.ndarray, groups: List[Tuple], alpha: float,
    Se: float) -> float:
    """
    Compute the equity-like index Z.
    Z = sum (1/(1-alpha)) * (1 - E_FN_group)^(1 - alpha)
    """
    Z = 0.0
    for (start, end) in groups:
        group_slice = risks[start:end]
        e_fn_group = e_fn(group_slice, Se)

        # Generalized alpha-fairness formula
        term = (1.0 - e_fn_group) ** (1.0 - alpha)
        Z += (1.0 / (1.0 - alpha)) * term

    return Z
```

Case $\lambda = 1$: Results with Real-World Data



Case $\lambda = 1$: Results with Artificial Data



Equity Analysis

Does Budget improve Fairness?

The results show that as the budget increases, the value of the objective function improves (decreases risk disparity).

- **Tight Budget:** The system is forced to group even medium-high risk subjects, creating disparity in False Negative risks.
- **Loose Budget:** High-risk subjects are isolated (Individual Testing). Since $E[FN]_{indiv} < E[FN]_{group}$, the "worst-off" subjects receive better care, improving the overall equity.

Conclusion: There is no conflict between Accuracy and Equity in this setting. Increasing resources improves both total detection (lower $E[FN]$) and fairness.

Case $\lambda < 1$

If $\lambda < 1$, the solution of the problem is no longer the same as the one obtained under the BM formulation. Moreover, we lose another key property that we have relied on so far, as:

- The optimal partition is not necessarily an ordered partition.

This leads to a much more complex problem, which is not solvable in polynomial time.

Furthermore, it can be possible that $PoF(\alpha) \neq 0$ for some value of α.

Heuristic Models

Heuristics: Motivation & Setup

Why heuristics?

- SM, BM, BM-E give optimal partitions but require solving shortest-path / MIP models.
- Heuristics are simple risk-based rules that a lab could implement without an optimizer.

Two policies.

1. **Equal-Size pooling (OD)**: one common pool size c.
2. **Threshold pooling (TOD-like)**: high-risk tested individually, low-risk pooled with size c.

Experimental setup.

- Dorfman two-stage testing with $Se = Sp = 0.95$.
- Risks generated by Monte Carlo from Table 2, populations of $N = 100$.
- Pool-size cap $M = 10$, following the literature.
- Same Dorfman formulas as SM to compute $\mathbb{E}[T(\Omega)]$ for a fair comparison.

Heuristic 1 — Equal-Size Pooling (OD)

Rule (Optimal Dorfman).

- Sort subjects by risk: $p_1 \leq \cdots \leq p_N$.
- For a given c, split into consecutive pools $\Omega_1, \Omega_2, \ldots$ of size c.
- For a pool Ω of size n:

$$\mathbb{E}[T(\Omega)] = \begin{cases} 1, & n = 1, \\ 1 + n\big[Se - (Se + Sp - 1)\prod_{i\in\Omega}(1 - p_i)\big], & n > 1. \end{cases}$$

- Total expected tests: $\mathbb{E}[T(c)] = \sum_{\Omega(c)} \mathbb{E}[T(\Omega)]$.

Choice of c. Exhaustive scan:

$$c^\star = \arg\min_{1\leq c\leq M} \mathbb{E}[T(c)].$$

Corresponds to **Optimal Dorfman (OD)** in McMahan et al. (2012).

Heuristic 2 — Threshold-Based Pooling (TOD-like)

Rule.

- Fix a cutoff τ and pool size c.
- Split the population:

$$H = \{i : p_i \geq \tau\}, \qquad L = \{i : p_i < \tau\}.$$

- Test H individually; pool L with equal-size groups of size c.

Expected tests.

$$\mathbb{E}[T(\tau, c)] = |H| + \sum_{\Omega \subset L} \mathbb{E}[T(\Omega)].$$

Calibration.

- Based on Thresholding OD (TOD) in McMahan et al. (2012).
- Grid search:

$$\tau \in \{0.002, 0.005, 0.01, 0.02, 0.05\}, \quad c \in \{1, \ldots, 10\}.$$

Comparison with SM (Minimizing $\mathbb{E}[T]$)

Monte Carlo design.

- 100 independent populations ($N = 100$ each) from Table 1.
- For each population:
  - **SM-optimal**: $\lambda_1 = \lambda_2 = 0$.
  - **Equal-Size OD**: best $c^\star$.
  - **Threshold TOD**: best $(\tau^\star, c^\star)$.

Average performance (Se = Sp = 0.95).

Method	mean $\mathbb{E}[T]$	std	min	max
Optimal SM	17.05	1.01	14.71	19.94
Equal-Size OD	18.15	1.39	14.71	21.52
Threshold TOD	18.93	1.82	14.71	24.09

Method	mean error	std	max
Equal-Size OD	6.34%	2.35%	10.36%
Threshold TOD	10.83%	5.20%	24.41%

Comparison with SM (Minimizing $\mathbb{E}[T]$)

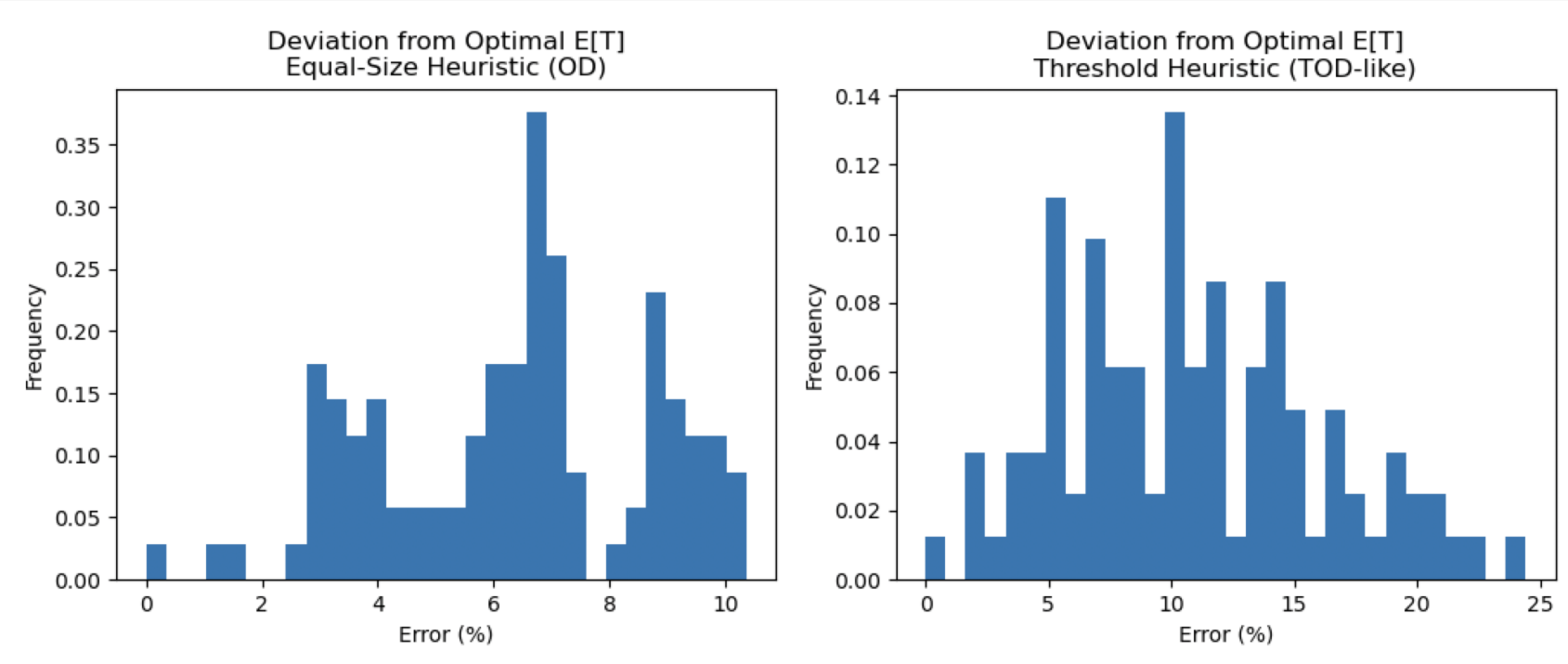


Figure 2: Deviation (in Percentage) from the Optimal Expected Number of Tests Obtained by SM for OD and TOD-like