Distribution Agreement

In presenting this thesis as a partial fulfillment of the requirements for a degree from Emory University, I hereby grant to Emory University and its agents the non-exclusive license to archive, make accessible, and display my thesis in whole or in part in all forms of media, now or hereafter now, including display on the World Wide Web. I understand that I may select some access restrictions as part of the online submission of this thesis. I retain all ownership rights to the copyright of the thesis. I also retain the right to use in future works (such as articles or books) all or part of this thesis.

Olin Gilster April 7 2026

Precedence-Aware Resource Allocation:

Extending the AUTOC Framework to Multi-Level Treatments

by

Olin Gilster

Dr. Shengpu Tang

Adviser

Department of Computer Science

Shengpu Tang

Adviser

Michelangelo Grigni

Committee Member

Ruoxuan Xiong

Committee Member

2026

Precedence-Aware Resource Allocation:

Extending the AUTOC Framework to Multi-Level Treatments

by

Olin Gilster

Shengpu Tang

Adviser

An abstract of

a thesis submitted to the Faculty of Emory College of Arts and Sciences

of Emory University in partial fulfillment

of the requirements of the degree of

Bachelor of Science with Honors

Department of Computer Science

2026

Abstract

Precedence-Aware Resource Allocation:
Extending the AUTOC Framework to Multi-Level Treatments

By Olin Gilster

Many real world decision problems require allocating limited resources across individuals to maximize total benefit often through interventions that vary in intensity rather than binary treat-or-not decisions. In these settings, higher-intensity treatments can require first treating lower levels of treatment, inducing a precedence constraint that complicates allocation under a fixed budget. While the causal inference literature has largely emphasized accurate estimation of conditional average treatment effects (CATE), recent work has shown that estimation accuracy alone does not guarantee effective resource allocation, motivating ranking-based evaluation frameworks. The area under the targeting operating characteristic (AUTOC) provides an effective way to evaluate prioritization rules across budget levels, but existing formulations are limited to binary treatments. We introduce Multi-Level AUTOC (ML-AUTOC), which is a generalization for AUTOC designed for multi-level treatment settings with precedence. We also propose precedence-aware policies for mapping true marginal treatment effects to feasible treatment assignments under fixed budget constraints. Our results show that in the multi-level treatment setting, maximizing realized treatment effects may require re-allocation of treatments across individuals.

Precedence-Aware Resource Allocation:
Extending the AUTOC Framework to Multi-Level Treatments

by

Olin Gilster

Dr. Shengpu Tang

Adviser

A thesis submitted to the Faculty of Emory College of Arts and Sciences

of Emory University in partial fulfillment

of the requirements of the degree of

Bachelor of Science with Honors

Department of Computer Science

2026

Acknowledgements

I would like to sincerely thank Dr. Shengpu Tang for his mentorship and guidance throughout this thesis. He consistently pushed me to think more rigorously, ask deeper questions, and strengthen both the intuition and formal foundations of my work. I am especially grateful for the opportunity to learn alongside him, as his curiosity and commitment to intellectual growth made this process both challenging and deeply rewarding.

Table of Contents

Chapter 1: Introduction **9**

1.1 Related Work: CATE 11

1.2 Related Work: AUTOC 12

Chapter 2: Problem Formulation **14**

2.1 Problem Statement 14

2.2 Solutions via Dynamic Programming 15

Chapter 3: Methods **17**

3.1 Scoring Functions, Rankings, and Allocation Policies 17

3.2 Naive Policies 19

3.3 Flattening the Multi-Level Allocation Problem 20

3.4 Greedy and Optimal Policies 22

3.5 Formalizing ML-AUTOC 24

Chapter 4: Research Question **27**

Chapter 5: Experiments and Results **28**

5.1 Datasets 28

5.2 RQ1: Strictly Increasing Marginal Benefit Dataset 31

5.3 RQ1: Random Marginal Benefit Dataset 32

5.4 RQ1: Strictly Decreasing Marginal Benefit Dataset 34

5.5 RQ1: Heterogeneous Strictly Increasing Marginal Benefit Dataset 36

5.6 RQ2: PAO versus IFP and PAG 38

Chapter 6: Discussion **39**

6.1 Precedence-Aware Optimal versus Precedence-Aware Greedy 39
6.2 Computational Complexity Analysis 40
6.3 When Does Flexibility Matter? 44
6.4 Non-Nested Allocations In The Clinical Setting 45
6.5 On The Use Of Non-Synthetic Data 46

Chapter 7: Conclusion **47**

Chapter A: Appendix **52**

A.1 Heterogeneous Treatment Costs 52
A.2 PGP, IFP, LFP, and PAG Code Implementation 56
A.3 Dynamic Programming for PAO Implementation 58
A.4 Dataset Generation 61

Chapter B: **64**

B.1 Run Time Data 64
B.2 Software 64

Chapter 1

Introduction

Across the domains of healthcare, education, and public-policy decision makers are often faced with the reality of resource constraints. For example, hospitals may find themselves confronting a surge of critically ill patients with limited number of beds or personnel. These constraints force hospitals to allocate their resources based on need, deciding who gets intensive care versus standard or minimal care [7, 11]. Beyond healthcare, other domains also face similar challenges. In education, overcrowded schools may limit the amount of attention each child gets. Thus school systems need to make informed decisions on who to assign excess teacher hours to maximize school test scores and student well-being [22]. In the public-policy setting, limited staffing at social outreach programs can reduce the amount of care given to each person. Specifically, in the context of housing, people may be assigned to increasingly better programs from basic shelter, transitional housing, and long-term assistance fully based on the amount of resources available to an organization [1].

The stakes of such allocation decisions are particularly acute in the healthcare setting [9]. Recent papers covering the COVID-19 pandemic found that when hospitals experienced a capacity strain, they had higher rates of mortality. Specifically, they found that in a nationwide resource constraint where ICU capacity was at 75%, "12,000 excess deaths would occur in the following two weeks." When ICU capacity was at 100%, it was predicted that 80,000 excess deaths would occur in the same time period [7]. Together, these examples highlight

the widespread challenge of allocating limited resources and the high cost of misallocation, emphasizing the need for systematic approaches to improve decision-making under resource constraints.

Researchers working at the intersection of causal inference and machine learning have largely focused on the problem of conditional average treatment effect (CATE) estimation, seeking to minimize the error between predicted and true individual treatment effects [15, 21]. Yet, accurate CATE estimation does not always translate to optimal resource allocation. To bridge this gap, researchers have increasingly pivoted toward ranking-based optimization and evaluation. For instance, Kamran et al. [12] proposed algorithms that directly optimize rankings for expected benefit, while Caniglia et al. [5] utilized dynamic marginal structural models (MSMs) to identify optimal strategies under resource usage constraints. To assess the quality of these prioritization rules independently of estimation accuracy, Yadlowsky et al. [23] popularized the area under the targeting operating characteristic (AUTOC) metric, previously described in Zhao et al. [24]. This metric quantifies how effectively a scoring function prioritizes individuals compared to a random allocation baseline across all possible budget levels.

Despite the utility of the AUTOC metric, existing works on ranking-based evaluation frameworks are limited to binary treatment decisions, where each individual is either treated or not treated. Current frameworks do not adequately address settings where interventions vary in intensity and follow a precedence structure, where higher-intensity treatments require first receiving lower-intensity ones. As this thesis will illustrate, naively converting such problems into a binary treatment formulation obscures these nuances and can lead to suboptimal allocation decisions. Therefore, extending ranking-based evaluation and decision rules beyond the binary setting is an essential building block for applying them to a larger range of real-world decision problems.

In this work, we propose a generalization of the binary AUTOC metric to settings involving multi-level treatments. This extension introduces unique challenges, primarily due to the

requirement of treatment level precedence. While a scoring function in a binary treatment setting maps to a unique ranking of individuals, our setting utilizes scoring functions that estimate marginal treatment effects for each individual at specific levels. Thus, each scoring function may correspond to multiple valid rankings of marginal treatment effects that respect precedence, and some that do not. Furthermore, the random allocation baseline must also be redefined to respect precedence. Finally, as resource budget increases, the optimal allocation may require re-allocating previously assigned resources to different individual-level pairs. This leads to the phenomenon of "flipping", which does not exist for binary treatment settings by definition. Specifically, this thesis makes the following contributions:

- We develop and formalize the multi-level AUTOC (ML-AUTOC) metric to evaluate prioritization efficiency of resource allocation tasks with multiple treatment levels.
- We introduce and compare several resource allocation rules, which we refer to as policies, that can be derived from scoring functions, and evaluate their ability to maximize realized treatment effects on synthetic datasets.
- We show that in multi-level treatment settings, our policies utilizing re-allocation fundamentally depart from ranking-based approaches, since maximizing realized treatment effects may require non-nested allocation policies that switch between partially treating one individual and initiating treatment for another.

1.1 Related Work: CATE

To understand the motivation for AUTOC and our extension to multi-level settings, it is important to examine the limitations of conditional average treatment effect (CATE) estimation in resource constrained problems. Consider the dataset $S = (x_i, t_i, y_i)_{i=1}^n$, where each individual i has covariates $x_i \in X \subseteq \mathbb{R}^d$, assigned treatment $t_i \in \{0, 1\}$, and observed outcome $y_i \in \mathbb{R}$. Under the potential outcomes framework [19], each individual is given potential outcomes $Y_i(0)$ and $Y_i(1)$ corresponding to control and treatment. Given these,

CATE is defined as:

$$\tau_i = \mathbb{E}[Y_i(1) - Y_i(0) \mid x_i].$$

CATE allows researchers to go beyond a single global estimate of the average treatment effect (ATE) by estimating τ_i conditioned on an individual's covariates x_i [15]. Despite the success in CATE research, evaluation of these methods often focuses on estimation accuracy rather than allocation efficiency [12, 23]. While CATE-based models can estimate heterogeneous treatment effects with high precision, they may still lead to suboptimal allocation decisions when resources are limited. It was found in Kamran et al. [12] that minimizing CATE predictive error does not necessarily translate to maximizing total utility under a budget constraint. Moreover, most CATE frameworks implicitly assume that treatment can be assigned without accounting for constraints on total resource use [5]. In practice, interventions often vary in cost and level, which require trade-offs where only a subset of individuals can be treated under a given budget. These limitations have motivated the development of ranking-based frameworks, such as AUTOC, which explicitly evaluates how effectively a model prioritizes individuals for treatment as available resources vary [12, 23].

1.2 Related Work: AUTOC

AUTOC provides the basis for evaluating treatment prioritization when the treatment structure is binary, and our proposed ML-AUTOC closely mirrors this framework in the multi-level setting; for this reason, it is important to first understand AUTOC.

The goal of AUTOC is to evaluate how well a scoring function $f(x_i)$, which can be mapped to a ranking over individuals, aligns with the true ranking induced by the conditional average treatment effect τ_i. Given a treatment budget $u \in [0, 1]$, $D_u^S(f)$ denotes the top u-fraction of individuals in S ranked the highest by score f:

$$D_u^S(f) = \{i \mid f(x_i) \geq \psi(f(x_i)_{i \in S}, u)\}$$

where $\psi(\cdot, u)$ represents the u-th percentile of model scores. The ATE at budget u induced by the scoring function f is defined as the mean of individual treatment effects for those individuals selected to be treated:

$$\mathrm{ATE}_u^S(f) = \frac{1}{|D_u^S(f)|} \sum_{i \in D_u^S(f)} \tau_i.$$

The targeting operating characteristic (TOC) at u quantifies how much better this ATE is than that of a random selection at the same budget. In the binary case, it is straightforward to obtain the random baseline since the expected ATE under random assignment is equal to the population ATE. Thus, the TOC is defined as:

$$\mathrm{TOC}_u^S(f) = \mathrm{ATE}_u^S(f) - \frac{1}{|S|} \sum_{i=1}^{|S|} \tau_i$$

Finally, the area under the TOC curve aggregates all budgets:

$$\mathrm{AUTOC}_S(f) = \frac{1}{|S|} \sum_{i=1}^{|S|} \mathrm{TOC}_{\frac{i}{|S|}}^S(f).$$

Intuitively, AUTOC captures the average improvement in treatment benefit obtained by using the model to evaluate patients at every possible resource level. A larger AUTOC value indicates that a ranking is more accurately prioritizing individuals who would benefit most from treatment, while an AUTOC = 0 is equivalent to random allocation [12, 23].

Chapter 2

Problem Formulation

2.1 Problem Statement

While AUTOC provides a strong evaluation metric for binary treatment allocation [1, 7, 22], many decision problems fall into *multi-level* intervention decisions where treatment intensity can be quantified as discrete levels $\ell \in \{0, 1, \dots L\}$. In our case, these treatment levels will exhibit precedence; that is, higher treatment levels may only be assigned if previous levels have been assigned. Each higher level typically provides a positive marginal benefit, denoted:

$$\delta_i^{(\ell)} = \mathbb{E}[Y_i(\ell) - Y_i(\ell - 1) \mid x_i] \tag{2.1}$$

representing the incremental gain from advancing an individual i from level $\ell - 1$ to ℓ. Standard AUTOC [12, 23, 24] which assumes a binary decision space $t_i \in \{0, 1\}$ cannot encapsulate these hierarchical relationships because it evaluates rankings on a single treatment basis.

Under this setting, consider the dataset $S = (x_i, t_i, y_i)_{i=1}^n$ where each individual i has covariates $x_i \in X$, assigned treatment levels $t_i \in \{0, 1, \dots L\}$, and observed outcome y_i.

Because of the treatment structure, each individual has a vector of potential outcomes:

$$Y_i = (Y_i(0), Y_i(1), \ldots, Y_i(L)),$$

which can be derived from the marginal treatment effects in Eq. (2.1). Given our goal is to evaluate rather than optimize a scoring function, in our experiments, we do not model or simulate covariates X directly. Instead, we directly simulate the outputs of the scoring functions in order to study how a given ranking should be translated into a treatment allocation policy under a fixed budget.

There remains a challenge in defining the budget for ML-AUTOC. In the standard AUTOC metric, the unit for each incremental movement in the budget is the choice to treat an individual. In our setting, individuals can be treated multiple times, which invalidates using them as a whole unit. To address this, we define the budget in terms of incremental treatment steps. Moving an individual from treatment level $\ell-1$ to level ℓ consumes one unit of budget, providing a natural and consistent unit of resource expenditure in the multi-level setting.

Our problem setting is closely related to multi-period precedence-constrained knapsack problems, which study the allocation of resources across hierarchical or ordered choices under a fixed budget. Prior work has explored such problems in static or multi-period settings (e.g., Aslan et al. [2], Moreno et al. [17], Samavati et al. [20]), typically focusing on optimizing a single objective value at a fixed budget.

2.2 Solutions via Dynamic Programming

The allocation problem underlying ML-AUTOC is closely related to the precedence-constrained multiple-choice knapsack problem (MCKP). Each individual i represents a class of mutually exclusive cumulative treatment options, corresponding to levels $\ell \in \{0, \ldots, L\}$, and each level produces cumulative benefit

$$V_i(\ell) = \sum_{k=1}^{\ell} \delta_i^{(k)} = \mathbb{E}[Y_i(\ell) - Y_i(0) \mid x_i]$$

Under the unit-cost assumption introduced in Section 2, assigning level ℓ to an individual consumes ℓ units of budget. The planner must choose at most one level per individual such that total budget does not exceed β and total benefit is maximized.

Formally, the problem for a fixed budget β is:

$$\max_{\ell_i \in \{0,\ldots,L\}} \sum_{i=1}^{n} V_i(\ell_i)$$

subject to

$$\sum_{i=1}^{n} \ell_i \leq \beta.$$

This formulation is a special case of the MCKP, one of the most well-studied combinatorial optimization problems [8, 13]. While Integer Linear Programming (ILP) formulations are common, dynamic programming (DP) methods provide a natural and computationally efficient approach when budgets are discrete and relatively small [16].

Dynamic programming is particularly well suited to our setting for two reasons:

1. **Evaluation Across All Budgets.** ML-AUTOC requires evaluating optimal allocations for every $\beta \in \{1, \ldots, nL\}$. A single DP table computes optimal values for all budgets simultaneously, whereas solving separate ILPs for each β would be redundant.

2. **Ranking Recovery.** The DP solution naturally stores transition decisions. By tracing which marginal increment is selected at each increase in β, we recover an implied ordering of treatment increments.

More information on the exact dynamic programming implementation can be found in Appendix A.3.

Chapter 3

Methods

3.1 Scoring Functions, Rankings, and Allocation Policies

Before describing our proposed approach, we need to distinguish a few related concepts that are equivalent under the binary treatment setting, but becomes tricky under multi-level treatments. In the binary treatment setting, a scoring function induces a unique ranking of individuals and treating the top ranked individuals at each budget defines an allocation policy. As a result, scoring functions, rankings, and allocation policies are all effectively interchangeable. In the multi-level setting, this equivalence no longer holds. Scoring functions assign values to individual-level pairs rather than individuals, and precedence constraints restrict what treatments are feasible at a given budget.

In the multi-level treatment setting, the goal is to construct a policy π that selects a set of individual–treatment level pairs (i, ℓ) subject to a budget constraint. Let

$$\Delta = \{(i, \ell) : i \in \{1, \ldots, n\},\ \ell \in \{1, \ldots, L\}\}$$

denote the set of all feasible individual–level pairs.

We define a policy as a mapping from a budget level $\beta \in \{0, 1, \ldots, nL\}$ to a feasible subset of pairs:

$$\pi : \{0, 1, \ldots, nL\} \to 2^{\Delta}, \quad \beta \mapsto \pi(\beta),$$

where $\pi(\beta) \subseteq \Delta$ contains the pairs selected by π under budget β.

As the budget increases, the sequence of sets $\{\pi(\beta)\}_{\beta}$ induces an ordering over treatment assignments. We interpret this sequence as a ranking over feasible individual–level pairs. To use our policies, we require a mechanism for prioritizing treatments as the budget increases. In the binary treatment setting, this is done by the scoring function $f(x_i)$, which induces an ordering over individuals. In the multi-level setting, this scoring can be defined following a similar idea in which we associate each treatment level ℓ with a level-specific scoring function $f_\ell(x_i)$. These scores are intended to rank individuals according to their predicted marginal benefit at level ℓ, $\delta_i^{(\ell)}$. Unlike in the binary case, these level-specific scores by themselves do not define how treatments should be allocated across levels and individuals as the budget increases. The best way to conceptualize the two objects, $f_\ell(x_i)$ and π, is to consider the scoring function first. The scoring function produces a level-specific ranking of treatment assignments based on predicted marginal benefits, which can be interpreted as a priority list. A policy then takes these rankings and applies a set of allocation rules, such as budget constraints, precedence requirements, or aggregation across levels, to determine which treatment assignments are ultimately selected. Importantly, this mapping is not unique. As we will demonstrate later in this section, it is possible that the same scoring function may give rise to multiple feasible policies and some allocation policies can not be represented as a single global ranking.

Consider the following multi-level treatment toy example (Table 3.1):

Table 3.1: True Marginal Benefit table

$\delta_i^{(\ell)}$	$\ell = 1$	$\ell = 2$
x_1	1	5
x_2	2	3

At level $\ell = 1$, since $\delta_2^{(1)} > \delta_1^{(1)}$, individual x_2 should receive priority for the first unit of treatment. However, at level $\ell = 2$, the ordering reverses, as $\delta_2^{(2)} < \delta_1^{(2)}$. This highlights a key challenge in the multi-level setting: **a single global score cannot, in general, correctly prioritize treatment assignments across all levels.** For instance, a global scoring rule of the form $f(x_i) \propto Y_i(L) - Y_i(0)$ (or equivalently $\sum_{\ell=1}^{L} \delta_i^{(\ell)}$) would rank $x_1 \succ x_2$ overall in this example ($6 > 5$). Under tight budgets, such a ranking would incorrectly allocate the first unit of treatment to x_1, despite x_2 having the larger marginal benefit at level $\ell = 1$.

3.2 Naive Policies

To convert scoring functions to policies in the multi-level treatment setting, one could naively convert the problem to a binary treatment problem. Below, we discuss three possibilities:

- Pooled Global Policy (PGP):

  This policy creates a global ranking by pooling all marginal scores $\{f_\ell(x_i) : \forall i, \ell\}$ across individuals i and levels ℓ, and selecting the highest values according to the marginal score ranking (e.g. 5, 3, 2, 1). While this identifies high-benefit treatment-level pairs, it frequently violates precedence constraints, for example, assigning Level 2 to a patient before they have received Level 1.

- Individual-First Policy (IFP)

  This approach aggregates an individual's total potential benefit, $\sum_\ell f_\ell(x_i)$, and ranks the individuals by this aggregate score (e.g., $1 + 5 = 6$, $2 + 3 = 5$). Resources are allocated to the top-ranked individual until they are fully treated across all levels before moving to the next individual (e.g., 1, 5, 2, 3). Within each individual treatment levels are assigned in increasing order of intensity, enforcing the precedence-induced ranking of levels. While this respects precedence, it cannot spread lower-level treatments across many individuals, even if the marginal benefits of Level 1 are significantly higher than the marginal benefits of Level 2.

- Level-First Policy (LFP): This policy imposes a strict hierarchy based on treatment levels, similar to classical level-based scheduling algorithms [10]. It requires that every individual in the population receive Level 1 before any individual can be considered for Level 2. Within each level, individuals are ranked by the marginal benefits of that level, for example, allocating Level 1 in the order (2, 1) and allocating Level 2 in the order (5, 3) for the total ranking of (2, 1, 5, 3).

3.3 Flattening the Multi-Level Allocation Problem

A key feature of the naive policies introduced above is that they reduce the dynamic multi-level allocation problem to a single static ranking problem. Rather than adaptively selecting treatments as the budget increases while respecting precedence constraints, these approaches construct a one-time global ordering over treatment actions. This effectively converts the multi-level decision problem into a binary-style allocation problem in which each treatment step is treated as an independent unit. We illustrate this flattening using the toy marginal benefit matrix in Table 3.1. Each policy maps this 2×2 matrix into a length-4 ranking over treatment actions (i, ℓ), effectively flattening the multi-level structure and computationally simplifying the problem.

1. PGP

   PGP ignores precedence at ranking time and globally sorts all (i, ℓ) pairs by their marginal benefit values. Sorting $\{1, 5, 2, 3\}$ in descending order yields the following flattened allocation matrix and ranking:

$$\begin{bmatrix} (x_1, 2) \\ (x_2, 2) \\ (x_2, 1) \\ (x_1, 1) \end{bmatrix}$$

2. IFP

   IFP first aggregates marginal benefits across levels for each individual,

$$\sum x_1 = 1 + 5 = 6, \qquad \sum x_2 = 2 + 3 = 5,$$

   and ranks individuals by this total score. Treatment actions are then expanded in precedence order for each individual. The resulting flattened allocation matrix and ranking is:

$$\begin{bmatrix} (x_1, 1) \\ (x_1, 2) \\ (x_2, 1) \\ (x_2, 2) \end{bmatrix}$$

3. LFP

   LFP sorts individuals independently within each treatment level and then concatenates the level-wise rankings. For $\ell = 1$, we have $x_2 \succ x_1$ since $2 > 1$; for $\ell = 2$, we have $x_1 \succ x_2$ since $5 > 3$. The resulting flattened allocation matrix and ranking is:

$$\begin{bmatrix} (x_2, 1) \\ (x_1, 1) \\ (x_1, 2) \\ (x_2, 2) \end{bmatrix}$$

In all three cases, the policy produces a fixed length-nL ranking prior to observing the budget. Therefore, allocation under any budget β is trivial because it corresponds to selecting the first β entries of the flattened list. The reduction in computational complexity to a binary problem explains both the computational efficiency and the corresponding limitations. Because their ordering is fixed ex ante, they cannot dynamically re-allocate in response to different marginal treatment structures.

It is important in this small example to consider the case of ties. In the fully synthetic experiments, marginal effects are drawn from continuous distributions so ties occur with a probability of 0. However, in practice ties may occur frequently. In these cases the choice of a tie breaking rule can materially effect the result. In particular, there is a distinction between allocating an additional treatment level to an already treated individual versus initiating treatment for a new individual. This introduces a local prioritization problem that is not captured by the flattened ranking and is not explored further in this work. In our experiments, ties were broken at random.

3.4 Greedy and Optimal Policies

Unfortunately, these policies come short as none of the them can fully capture the structure of the problem: they either ignore how marginal benefits vary by level or fail to respect the hierarchical dependencies across treatment levels. What we want is a policy that (i) accounts for level-specific marginal benefits and (ii) preserves precedence constraints.

Because the PGP, IFP, and LFP policies fail to balance precedence with marginal utility, we propose two precedence-aware policies that dynamically navigates these trade-offs to maximize total population benefit.

A feasible allocation must satisfy the precedence constraint that, for any individual i, treatment level ℓ can only be assigned if all lower levels have already been assigned. Formally, if (i, ℓ) is selected, then (i, k) must also be selected for all $k < \ell$. Once taking into account what possible treatments are available at a given time, we will use a greedy strategy to select the one with the highest marginal benefit as the next one to treat. Two possible ranking structures can be built off of these rules, Precedence-Aware Greedy (PAG) and Precedence-Aware Optimal (PAO). The distinction between the two rankings lies in the reallocation of treatments at any distinct budget. The PAG policy constructs the ranking incrementally, fixing earlier treatment assignments and never revising past decisions as the

budget expands. In contrast, the PAO policy allows re-optimization at each budget level, potentially revising earlier allocations to account for newly feasible treatment options. From an algorithmic perspective, PAO corresponds to solving a sequence of budget-constrained optimization problems via dynamic programming. Although we will not discuss it here, a detailed description of the dynamic programming formulation and its computational complexity is provided in Appendix A.3.

For the toy example in Table 3.1, each policy is evaluated compared to a random baseline allocation under the same budget, which is defined in Section 3.5. In this specific case, all feasible sets were calculated to determine a true average. Following these ranking definitions, we can plot the TOC curves for each policy, showing budget levels versus average benefit across the cohort:

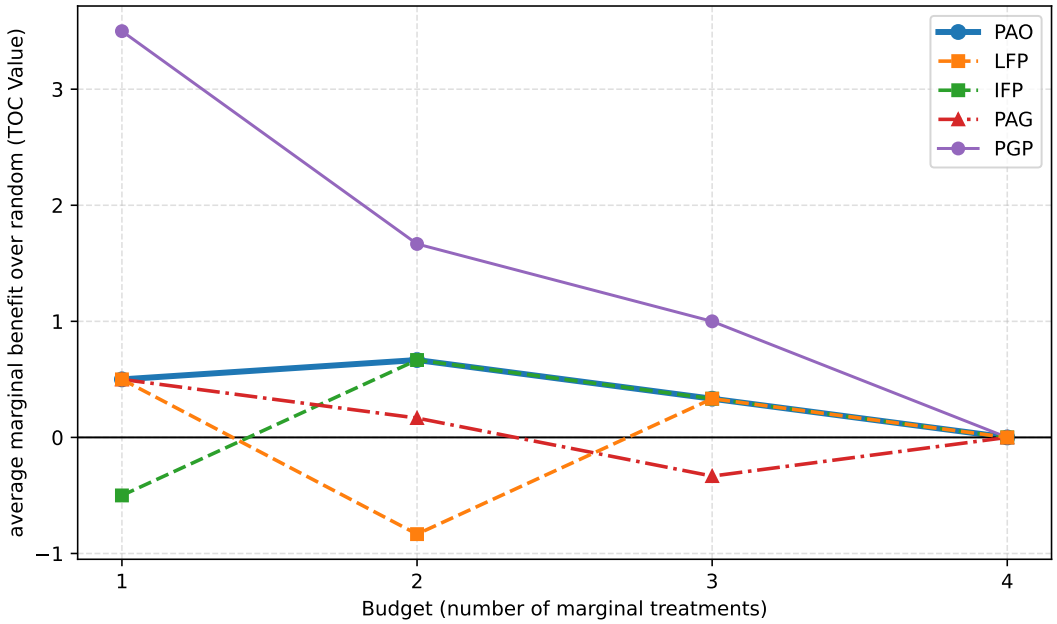

Figure 3.1: Toy example ML-TOC Graph

The TOC curves in Fig. 3.1 illustrate how different ranking policies behave under the same non-binary marginal benefit structure, demonstrating how enforcing precedence constraints can alter the ordering of policy performance.

3.5 Formalizing ML-AUTOC

Because individuals can be treated incrementally across multiple levels, defining treatment thresholds in terms of the top u-fraction of individuals is not a meaningful metric. Instead, we index allocations by a scalar budget parameter $\beta \in \{0, 1, \ldots, nL\} = \mathcal{B}$, which represents the total number of treatment units assigned across all individuals and levels.

Let $f = \{f_\ell : \ell = 1, \ldots, L\}$ denote the collection of level-specific scoring functions. We define the allocation set under budget β as

$$D_\beta^S(f) = \pi_f(\beta), \tag{3.1}$$

where $\pi_f(\beta) \subseteq \Delta$ is the set of individual–level pairs selected by the policy induced by f under budget β. The average marginal benefit realized under budget β is then:

$$\mathrm{ATE}_\beta^S(f) = \frac{1}{|D_\beta^S(f)|} \sum_{(i,\ell) \in D_\beta^S(f)} \delta_i^{(\ell)}. \tag{3.2}$$

The targeting operating characteristic at budget β quantifies how much better the selected subset is compared to a baseline equivalent to random selection and needs to respect the treatment level precedence structures. To calculate the random baseline, we will define $\mathcal{F}_\beta$ as the set of all feasible allocations of individual-level pairs under a budget β that satisfy precedence. We consider a random allocation $\widetilde{A}_\beta$ drawn uniformly from $\mathcal{F}_\beta$. Therefore, for all $(i, \ell) \in \widetilde{A}_\beta$ we can define RAND_β as the $\mathbb{E}_{\widetilde{A}_\beta}[\sum_{(i,\ell)\in\widetilde{A}_\beta} \delta_i^{(\ell)}]$. Now for a scoring function that induces a feasible allocation, TOC can be defined as:

$$\mathrm{TOC}_\beta^S(f) = \mathrm{ATE}_\beta^S(f) - \mathrm{RAND}_\beta. \tag{3.3}$$

Let $\mathcal{B} \subseteq \{1, \ldots, nL\}$ denote the set of feasible budgets. The term $|\mathcal{B}|$ denotes the cardinality so $\frac{1}{|\mathcal{B}|}$ represents the division by these budgets. Finally, the ML-AUTOC score aggregates

realized average treatment effects across all budgets:

$$\text{ML-AUTOC}_S(f) = \frac{1}{|\mathcal{B}|} \sum_{\beta \in \mathcal{B}} \text{TOC}_\beta^S(f). \tag{3.4}$$

ML-AUTOC and AUTOC are designed to measure the same conceptual object, which is how well a learned scoring rule prioritizes units for treatment as resources increase. The difference is in how the comparison is made. AUTOC is defined for binary treatment, where each additional unit of budget corresponds to treating one new individual and the random baseline is a uniform draw over subsets of individuals of a given size. In ML-AUTOC, the budget expands over treatment increments (i, ℓ) which are subject to precedence, so the random baseline must be defined over the feasible allocation set. This change in the feasible set (and therefore in the baseline being subtracted) is the fundamental difference between AUTOC and ML-AUTOC.

A second difference is the unit for which the area is accumulated. In binary AUTOC, the horizontal axis is naturally interpreted as the treated fraction u. In the multi-level setting, the natural budget unit is the number of incremental level assignments, so the curve aggregates performance over $\beta = nL$ possible budget increments. One could equivalently normalize by the maximum budget nL, expressing performance as a function of the fraction of budget expended. This normalized representation is used in our plots, where the horizontal axis corresponds to the percentage of budget. One could alternatively normalize the horizontal axis by the number of unique individuals treated at each budget, yielding a per-patient notion of efficiency. This normalization is not effective because it is not aligned with the resource model in which costs increase per incremental level, so it would conflate different allocations that treat the same number of individuals but at different intensities.

Finally, ML-AUTOC has the same value interpretation as AUTOC for policies that induce a nested ranking over treatment actions, and can be extended as a cumulative performance metric for more general allocation policies. Naive policies such as LFP, IFP, and PGP as

well as PAG produce an explicit ordering of (i, ℓ) actions, so the ML-AUTOC area can be interpreted as a cumulative comparison to the random precedence-feasible baseline as the budget expands. Although PAO does not generate a single nested ranking because it re-optimizes the allocation at each budget and may revise earlier decisions, its ML-AUTOC can still be computed as the cumulative average treatment effect relative to the random baseline across budgets. In this sense, ML-AUTOC provides a unified metric for comparing both nested and non-nested policies. In addition to ML-AUTOC, we also compare policies to PAO budget-wise by reporting the fraction of budgets for which one policy achieves a higher realized benefit than another ("win %"). This complementary metric highlights pointwise performance differences that may not be fully captured by cumulative area comparisons.

Chapter 4

Research Question

Does a precedence-aware greedy policy achieve higher total average treatment effect (ATE) than naive ranking policies in multi-level treatment settings with precedence constraints?
To investigate this, we examine the following empirical questions:

1. **Policy Performance:** On synthetic datasets with different per-level treatment effect structures, how do PAG and PAO compare to naive polices in terms of realized total average treatment effect across budget levels?

2. **Structural Behavior:** How do allocations produced by PAO which allows for re-allocations differ from those produced by nested greedy allocations from PAG, and when does relaxing the monotonic allocation requirement lead to a higher realized average treatment effect?

Chapter 5

Experiments and Results

5.1 Datasets

In order to evaluate ML-AUTOC under varying structural conditions, we generated 100 synthetic datasets of four types that were designed to isolate how different marginal benefit structures influence policy performance. Synthetic data was used for our experiments following the reasoning from Kamran et al. [12] that true individual-level treatment effects are rarely observable in practice. Since only one realized outcome is observed per individual, evaluating optimal treatment policies is infeasible using real data. Moreover, our framework assumes access to marginal treatment benefits, which further motivates the use of synthetic environments to simulate ground-truth performance under controlled assumptions.

Each dataset consists of (n) individuals, where treatment is administered across levels $\ell \in 1, \ldots, 4$ (i.e., $L = 4$). Advancing an individual from level $\ell - 1$ to ℓ incurs one unit of budget. Thus, any differences in model performance arise solely from the marginal benefit structure and from how rankings interact with precedence constraints. The datasets below were constructed to isolate how different marginal benefit structures interact with precedence constraints and to evaluate both the performance of ranking policies in terms of realized treatment effect. In addition, their design facilitated the study of when naive, nested rank-

ing heuristics fail and when precedence-aware, budget-adaptive policies can achieve higher realized benefit. In particular, they allowed us to:

- RQ1: evaluate how the proposed PAG and PAO policies compare to naive policies in terms of realized treatment effect across budgets,
- RQ2: evaluate when relaxing nested (prefix-based) ranking requirements, by allowing reallocation across budgets in PAO, leads to a higher realized total treatment effect compared to the nested greedy PAG policy.

The full data-generating process, including distributional assumptions and implementation details for each dataset type, is provided in Appendix A.4.

Strictly Increasing Marginal Benefit Dataset. In this dataset, higher treatment levels are always substantially more beneficial. To generate this, we first draw a baseline benefit for each level and each individual from a normal distribution. We then enforce monotonicity by increasing each successive level's marginal benefit by at least one unit if the raw draw does not satisfy this. As a result, the marginal benefits always satisfy: $\delta_i^{(\ell)} \geq \delta_i^{(\ell-1)} + 1$.

Random Marginal Benefit Dataset. In this dataset, marginal benefits are nonnegative and have no ordering across treatment levels. For each individual i and level ℓ, the marginal benefit $\delta_i^{(\ell)}$ satisfies $\delta_i^{(\ell)} \geq 0$, but is otherwise unconstrained across levels. In particular, no monotonicity condition is imposed: higher treatment levels may yield greater, smaller, or comparable marginal benefit relative to earlier levels. As a result, individuals may benefit most from early treatment intensities, later intensities, or intermediate levels, and these patterns can vary irregularly across both individuals and levels.

Strictly Decreasing Marginal Benefit Dataset. In this dataset, marginal benefits are strictly decreasing across treatment levels, so earlier treatment intensities are always more beneficial than later ones. This dataset is constructed by taking the Strictly Increasing Marginal Benefit Dataset that appears as follows:

$$\delta_i^{(1)} < \delta_i^{(2)} < \delta_i^{(3)} < \delta_i^{(4)}$$

and applies a transformation so that:

$$\tilde{\delta}_i^{(1)} = \delta_i^{(4)}, \quad \tilde{\delta}_i^{(2)} = \delta_i^{(3)}, \quad \tilde{\delta}_i^{(3)} = \delta_i^{(2)}, \quad \tilde{\delta}_i^{(4)} = \delta_i^{(1)}.$$

where $\tilde{\delta}_i^{(\ell)}$ denotes the marginal benefit at level ℓ in the strictly decreasing dataset. Formally, the marginal benefits are constrained to satisfy: $\delta_i^{(\ell)} \leq \delta_i^{(\ell-1)} - 1$ for all individuals and levels.

Heterogeneous Strictly Increasing Marginal Benefit Dataset. In this dataset, marginal benefits remain strictly increasing across treatment levels for each individual, but the scale and distribution of these benefits vary across individuals. Unlike the homogeneous strictly increasing setting considered above, individuals are not drawn from a common marginal benefit distribution. Instead, each individual i has their own scale parameter that governs the magnitude of their marginal benefits.

To construct this dataset, we first draw a base benefit for each individual from a distribution (e.g., a log-normal distribution). Conditional on this base value, marginal benefits are generated sequentially across levels, enforcing the strictly increasing constraint:

$$\delta_i^{(\ell)} \geq \delta_i^{(\ell-1)} + 1.$$

However, because individuals differ in their baseline scale, it is possible for lower treatment levels of one individual to exceed higher treatment levels of another. That is, for individuals i and j, it may occur that

$$\delta_i^{(2)} > \delta_j^{(3)}.$$

This construction introduces cross-individual heterogeneity while preserving within-individual monotonicity. As a result, the dataset captures environments in which treatment intensification remains beneficial for each individual, but the relative ranking of marginal increments across individuals and levels is nontrivial.

This setting is particularly useful for evaluating how policies handle trade-offs between completing treatment for high-potential individuals versus allocating early treatment levels across many individuals. It therefore provides a controlled environment to study when precedence-aware policies outperform ranking-based heuristics that do not account for cross-individual heterogeneity.

5.2 RQ1: Strictly Increasing Marginal Benefit Dataset

As a result of the structure of the strictly increasing dataset, policies that prioritize allocating higher levels early achieve a substantially higher average treatment effect across budgets. Among policies that respect the precedence structure (i.e., all policies except PGP) , both IFP and PAG achieve similar performance. To further illustrate how policies behave across the full budget range, we examine the average ML-TOC curves aggregated across all datasets. Fig. 5.1 plots the mean targeting operating characteristic for each policy, with budgets on the horizontal axis and average treatment effect relative to a random baseline on the vertical axis. Consistent with the aggregate results in Table 5.1, the average curves reveal that IFP, PAG, and PAO exhibit nearly identical performance across all budget levels.

Table 5.1: ML-AUTOC Values for Strictly Increasing Marginals

Policy	ML-AUTOC (mean ± std)
PGP	1.666352 ± 0.003744
IFP	0.941827 ± 0.009646
LFP	-0.154488 ± 0.003209
PAG	0.940726 ± 0.009646
PAO	0.941828 ± 0.009646

The LFP curve remains distinctly separated from the other policies, not only due to its lower overall performance, but also because of its characteristic shape. This pattern reflects the fact that LFP exhausts the highest marginal benefits at early budgets by fully allocating lower treatment levels across individuals before advancing to higher levels.

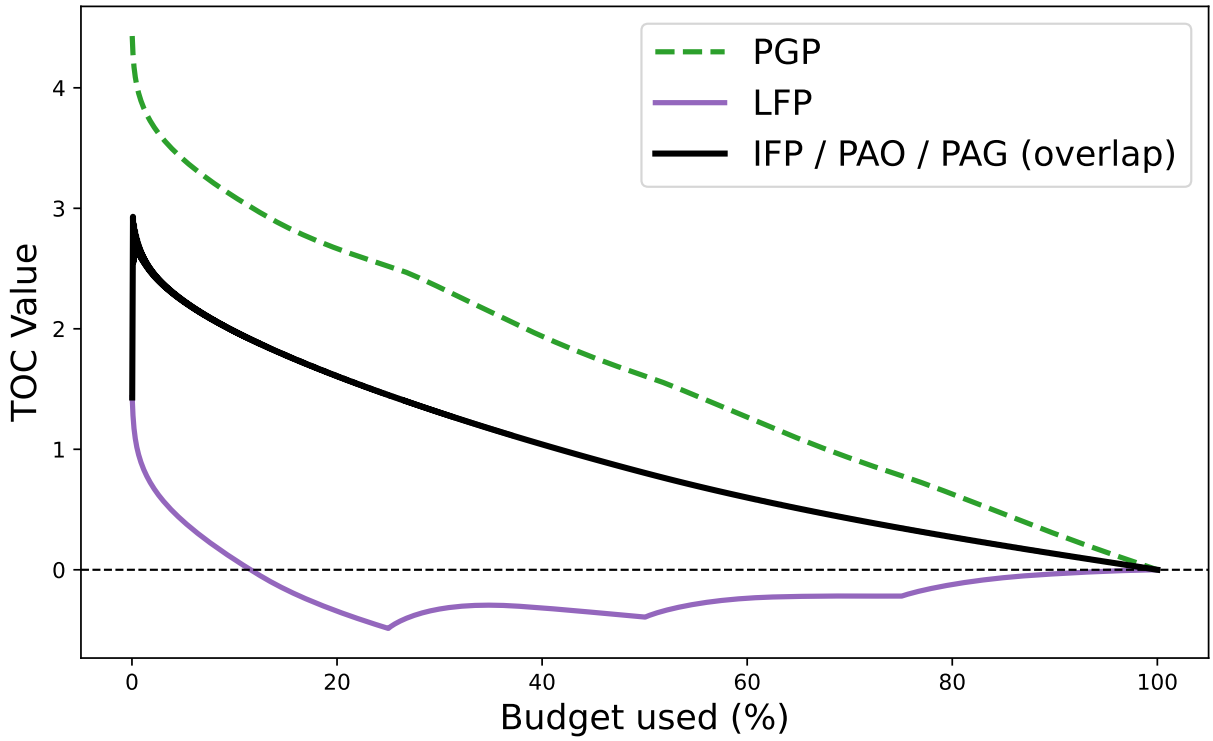

Figure 5.1: Average TOC curves across budgets for the strictly increasing marginal benefit dataset. Curves are averaged over 100 datasets and plotted relative to a random baseline. ± one standard deviation error bars are plotted but too small to be visually distinguishable.

Overall, in settings with strictly ordered marginal benefits, these findings indicate that allowing flexible, non-nested allocation (as in PAO) provides no additional benefit in strictly increasing settings, where the optimal allocation is inherently nested and can be naively achieved by IFP.

5.3 RQ1: Random Marginal Benefit Dataset

Unlike in the strictly increasing case, in the random marginal case clear separation emerges between policies as the budget increases. The flexible precedence-aware policy PAO consistently dominates all other valid policies.

While the greedy policy PAG initially outperforms the global score policy IFP, the two curves intersect at approximately 25% of the total budget ($\beta \approx 1000/4000$). Beyond this

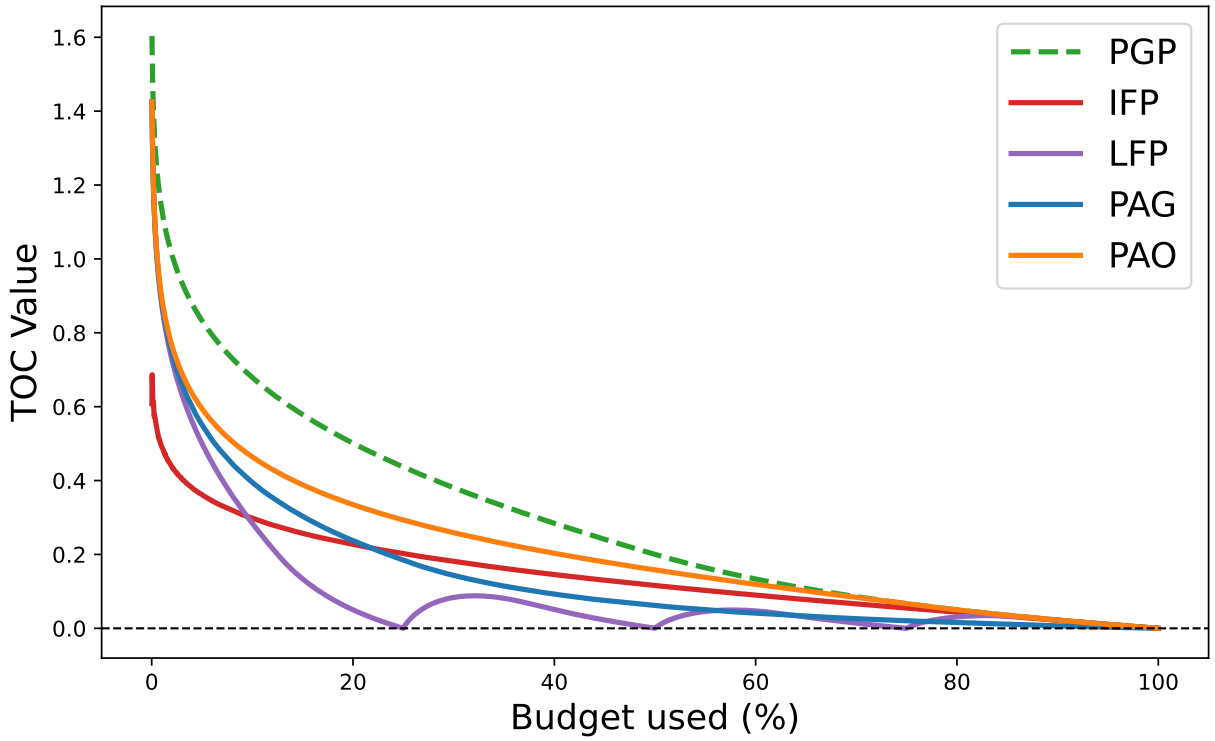

Figure 5.2: Average TOC curves across budgets for the random marginal benefit dataset. Curves are averaged over 100 datasets and plotted relative to a random baseline. ± one standard deviation error bars are plotted but too small to be visually distinguishable.

point, IFP achieves higher realized average treatment effect for the remainder of the budget range. This crossover highlights the cost of enforcing nested (prefix-based) rankings when marginal benefits vary across treatment levels, as early allocation decisions restrict the ability of PAG to adapt at later budgets. This suggests that in the case of smaller budgets and random marginal benefits, PAG may be a better policy to pursue to allocate resources.

In contrast, PAO avoids this tradeoff entirely by allowing non-nested, dynamically optimized allocations, enabling it to select the highest marginal benefit action at each budget. As a result, PAO maintains superior performance across nearly all budget levels.

Between precedence-respecting policies, PAG and IFP achieve nearly identical ML-AUTOC values, despite exhibiting different behavior across the budget range in Fig. 5.2. This highlights an important distinction between pointwise performance and cumulative performance:

although IFP overtakes PAG at larger budgets, the early-budget advantage of PAG is sufficient to equalize total realized benefit when aggregated across all budgets.

However, PAO achieves a substantially higher ML-AUTOC than both PAG and IFP, indicating that flexible allocation improves not only pointwise performance but also cumulative performance when marginal benefits are heterogeneous.

Table 5.2: ML-AUTOC Values for Random Marginals

Policy	ML-AUTOC (mean ± std)
PGP	0.285943 ± 0.005926
IFP	0.141575 ± 0.004947
LFP	0.096445 ± 0.003341
PAG	0.141024 ± 0.004727
PAO	0.209599 ± 0.004183

Overall, in the random marginal benefit setting, relaxing nested ranking constraints enables higher pointwise performance across budgets, while cumulative ML-AUTOC comparisons reveal tradeoffs between early- and late-budget performance among nested policies.

5.4 RQ1: Strictly Decreasing Marginal Benefit Dataset

In the strictly decreasing marginal benefit setting, PGP, PAG, PAO, and LFP exhibit nearly identical performance. This occurs because optimal allocations in this setting prioritize exhausting high early marginal benefits, leaving little room for gains from reallocating treatment across individuals at later budgets. IFP performs worse because it is forced to treat an entire individual before moving on to the next and therefore is unable to capitalize on larger marginals earlier. As a result, flexibility provides no meaningful advantage when marginal benefits are strictly front-loaded.

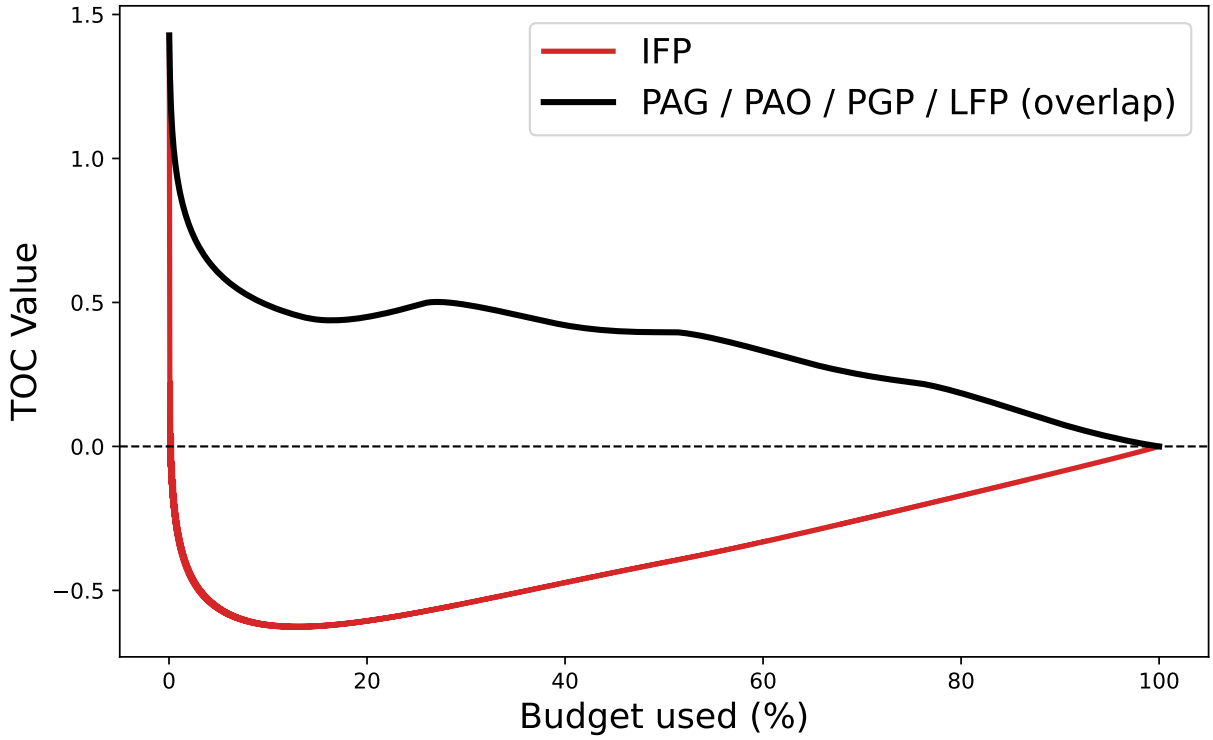

Figure 5.3: Average TOC curves across budgets for the strictly decreasing marginal benefit dataset. Curves are averaged over 100 datasets and plotted relative to a random baseline. ± one standard deviation error bars are plotted but too small to be visually distinguishable.

Table 5.3: ML-AUTOC Values for Strictly Decreasing Marginals

Policy	ML-AUTOC (mean ± std)
PGP	0.350069 ± 0.003820
IFP	-0.368648 ± 0.009757
LFP	0.348233 ± 0.003348
PAG	0.350069 ± 0.003820
PAO	0.350069 ± 0.003820

5.5 RQ1: Heterogeneous Strictly Increasing Marginal Benefit Dataset

The heterogeneous strictly increasing marginal benefit setting introduces substantial overlap in the ranking of marginal treatment increments by PAO, PAG, and IFP. The ML-AUTOC values in Table 5.4 show that IFP, PAO, and PAG achieve nearly identical performance. The differences across these policies are negligible both in mean performance and variance, indicating that the additional flexibility of PAO does not translate into meaningful gains.

Table 5.4: ML-AUTOC Values for Heterogeneous Strictly Increasing Marginals

Policy	ML-AUTOC (mean ± std)
PGP	1.901681 ± 0.019583
IFP	1.394540 ± 0.028555
LFP	0.003476 ± 0.008919
PAG	1.392382 ± 0.028573
PAO	1.394548 ± 0.028554

To further illustrate how these policies behave across the full budget range, we examine the average ML-TOC curves averaged across datasets. Fig. 5.4 plots the TOC for each policy, with budgets on the horizontal axis and average treatment effect relative to a random baseline on the vertical axis. Consistent with the results in Table 5.4, the curves for IFP, PAO, and PAG overlap and are visually indistinguishable across all budget levels.

This overlap indicates that, despite cross-individual heterogeneity, all three policies in-

duce nearly identical rankings of marginal treatment increments. While heterogeneity introduces crossings in the global ordering, it does not fundamentally disrupt the prefix structure of the optimal allocation. As a result, policies that enforce nested allocations remain sufficient to capture the majority of attainable gains.

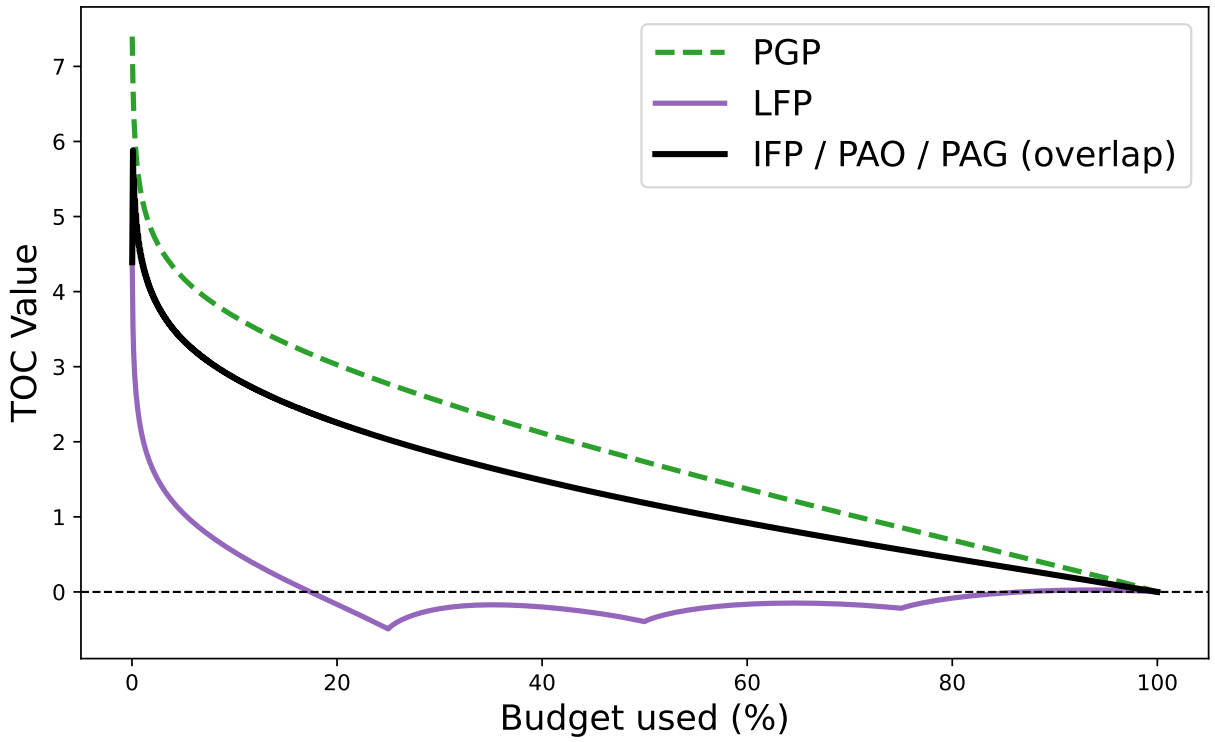

Figure 5.4: Average TOC curves across budgets for the heterogeneous strictly increasing marginal benefit dataset. Curves are averaged over 100 datasets and plotted relative to a random baseline. $\pm$ one standard deviation error bars are plotted but too small to be visually distinguishable.

5.6 RQ2: PAO versus IFP and PAG

Table 5.5 quantifies the structural differences between flexible and nested precedence-aware policies by measuring how often PAO strictly outperforms competing policies across budgets.

Table 5.5: Frequency with which the flexible precedence-aware policy PAO strictly outperforms baseline policies across budgets.

Dataset	% Budgets PAO > IFP	% Budgets PAO > PAG
Strictly Increasing	45.41 ± 31.32	87.65 ± 7.82
Random	96.70 ± 3.15	99.55 ± 0.26
Strictly Decreasing	99.94 ± 0.02	44.44 ± 29.59
Heterogeneous Increasing	47.92 ± 26.15	94.17 ± 3.83

In settings with strictly increasing marginal benefits, allowing reallocation across budgets yields frequent improvements over PAG, but does not translate into consistent gains over IFP. In contrast, when marginal benefits are random and heterogeneous across levels, relaxing the nested (prefix-based) constraint enables PAO to consistently dominate both IFP and PAG across nearly all budget levels. In this regime, the ability to reallocate across individuals and levels is critical, as the globally optimal sequence of marginal increments does not follow a simple prefix structure. When marginal benefits are strictly decreasing, PAO almost always outperforms IFP but offers no systematic advantage over PAG, reflecting the fact that optimal allocations in this setting favor concentrating resources early rather than flexibly reallocating across individuals. In this case, the optimal policy is naturally nested, so flexibility provides little additional benefit. Finally, in the heterogeneous strictly increasing setting, the conclusions are similar to the strictly increasing setting. Allowing reallocation in this setting does lead to consistent improvements over PAG at individual budgets but not over IFP.

Chapter 6

Discussion

6.1 Precedence-Aware Optimal versus Precedence-Aware Greedy

As mentioned in Section 3.4, PAO is uniquely different from the other policies in that it does not have a direct interpretable ranking that can be derived from the policy allocation. Across both datasets, PAO outperforms PAG at the majority of budget levels, with particularly strong dominance in the random marginal benefit setting. This improvement comes at the cost of interpretability. Because allocations produced by PAO are not nested, following the policy at budget β does not guarantee that the treatment chosen at budget $\beta + 1$ extends the previous allocation in a greedy manner. In particular, maximizing realized average treatment effect may require reallocating budget away from partially treated individuals toward initiating treatment for others, breaking the prefix structure that underlies ranking-based approaches such as PAG.

When individuals can receive multiple treatment intensities, optimal allocation may require switching between individuals and treatment levels as the budget evolves, rather than committing to a fixed ordering. PAO captures this flexibility, demonstrating that in multi-level intervention problems, maximizing realized treatment effect can necessitate departing

from nested ranking structures that are sufficient in the binary AUTOC setting.

6.2 Computational Complexity Analysis

To evaluate the computational cost of each policy, we measure the runtime of computing each ranking on a strictly increasing marginal benefit dataset of depth $L = 4$ and varying population size n. Section 6.2 and Table 6.1 report the time required to generate a full policy at each size n.

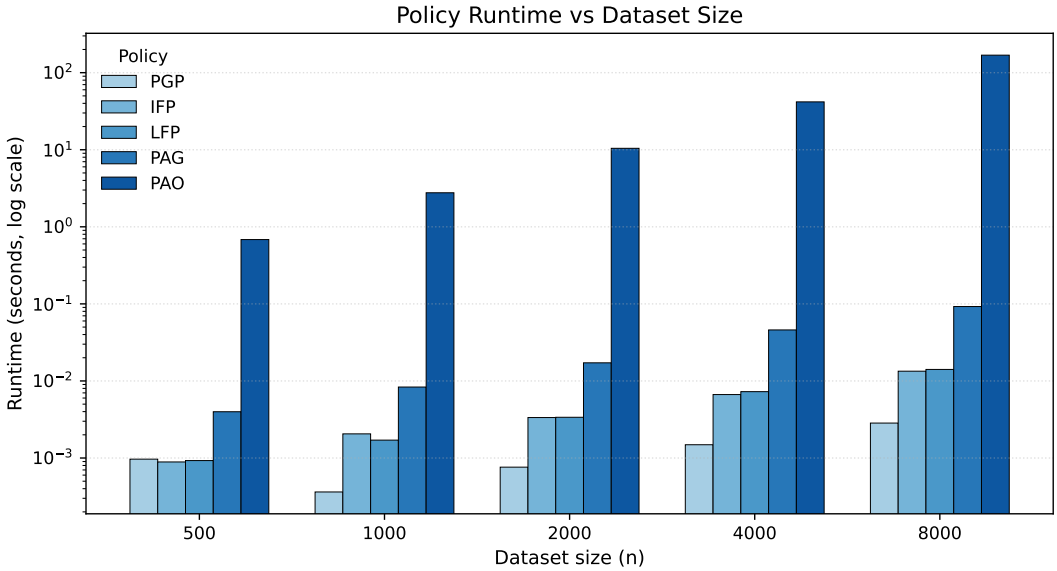

Figure 6.1: Empirical runtime scaling of ranking policies as the number of individuals n increases, measured on strictly increasing marginal benefit datasets with $L = 4$.

The naive ranking policies of LFP, IFP, and PGP exhibit negligible runtimes across the dataset sizes considered. While these policies differ in how rankings are constructed, each reduces the multi-level allocation problem to a small number of comparison-based sorting operations, with the number of sorts bounded above by the treatment depth L. As a result, their computational cost is dominated by standard sorting steps whose total complexity scales near-linearly in the number of individuals. For fixed treatment depth L, this leads to

effective runtimes that grow approximately on the order of $\mathcal{O}(n \log n)$, consistent with the empirical timing results shown in Section 6.2.

Between PAG and PAO, PAG consistently achieves a lower runtime. This difference is substantial, with PAG running orders of magnitude faster than PAO in practice. This gap arises from the dynamic programming procedure used by PAO to compute a globally optimal allocation independently at each budget level. While Section 6.2 showed that this flexibility can improve solution quality, it introduces *substantial* computational overhead, limiting its usefulness.

These results highlight a clear tradeoff between computational efficiency and allocation flexibility. While PAO provides the most reactive policy by re-allocating every budget level, its runtime makes it impractical for large-scale cases. In contrast, PAG scales comparably to the naive ranking policies while still enforcing precedence constraints. As a result, PAG likely is the preferred policy in applications where computational constraints are binding, and rankings must be generated at scale.

Table 6.1: Asymptotic runtime of ranking and policies as a function of the number of individuals n and treatment levels L.

Policy	**Runtime (Big-Oh)**
PGP	$O(nL \log(nL))$
IFP	$O(nL + n \log n)$
LFP	$O(nL \log n)$
PAG	$O(nL \log n)$
PAO	$O(n^2 L^2)$

We analyze the computational complexity of the naive ranking policies as a function of the number of individuals n and treatment levels L. Our budget β represents our total treatment actions. Our analysis relies on well-known results for comparison-based sorting that establish a worst-case lower bound of $\Omega(k \log k)$ comparisons for sorting k keys, with matching $O(k \log k)$ upper bounds achievable by standard algorithms [14].

1. PGP

   The implementation of PGP in Appendix A.2 first flattens the $(n \times L)$ marginal benefit matrix into a length nL vector, which costs $O(nL)$.

   The policy then sorts these marginal benefit values, which costs $O(nL \log(nL))$ under comparison-based sorting. Constructing the ranking array from the sorted indices requires writing nL entries and therefore costs $O(nL)$. The sorting step dominates, so the total running time of PGP is $O(nL \log(nL))$.

2. IFP

   The implementation of IFP in Appendix A.2 first computes an aggregate score for each individual by summing marginal benefits across all levels. This requires iterating over the full $(n \times L)$ matrix once, costing $O(nL)$.

   The policy then sorts the n aggregate scores, which costs $O(n \log n)$ under comparison-based sorting. Finally, the nested loop expands each individual into its L treatment actions and writes nL entries into the ranking array, costing $O(nL)$. Combining terms yields a total running time of

   $$O(nL + n \log n).$$

3. LFP

   The implementation of LFP in Appendix A.2 consists of an outer loop over levels $\ell = 1, \ldots, L$.

   The policy computes a sort for each level which costs $O(n \log n)$ under comparison-based sorting. The subsequent inner loop iterates once over the sorted indices and writes n entries into the ranking array, costing $O(n)$. Summing over all levels yields $O(L\, n \log n)$. Since the array writes are asymptotically dominated by the sorting step, the total running time of LFP is:

   $$O(nL \log n)$$

4. PAG

   The implementation of PAG consists of selecting treatment actions sequentially over the full budget horizon of nL marginal increments. Rather than scanning all n individuals at each step, the algorithm maintains the current feasible frontier in a priority queue. Initially, the queue contains the first feasible marginal benefit for each of the n individuals.

   At each budget step, the algorithm removes the maximum feasible marginal benefit from the priority queue, which costs $O(\log n)$. If the selected action corresponds to level ℓ for individual i, then level $\ell + 1$ for that same individual becomes newly feasible and is inserted into the priority queue, which also costs $O(\log n)$. Thus, each budget step requires at most one extract-max and one insert operation, for a total cost of $O(\log n)$ per step.

   Since there are nL total treatment actions, the total running time is

$$O(nL \log n).$$

   The initialization of the priority queue with one entry per individual costs $O(n)$, which is dominated by the main loop. Therefore, the overall running time of the implementation is

$$O(nL \log n).$$

5. PAO

   The implementation of PAO first computes prefix sums across treatment levels for each individual, which costs $O(nL)$. The policy then applies a dynamic program over individuals and budgets nL, updating an array of length nL for each of L possible allocation sizes per individual. This yields a total dynamic programming cost of $O(nLnL)$.

Therefore, the overall running time of PAO is

$$O(n^2L^2).$$

6.3 When Does Flexibility Matter?

Our experiments suggest that the value of flexibility in allocation relies on the structure of marginal benefits. Specifically, to what extent the optimal decision at multiple budget levels β in a cohort requires switching treatments.

In the strictly increasing marginal benefit setting (heterogeneous and homogeneous), treating higher levels as early as possible is the most beneficial to overall score. This intuition is verified by our empirical observation that several policies exhibited the same behavior: treating an individual fully once started. It was also observed that this behavior reduced the importance of reallocation as treating a higher level provided more benefit than reallocating. Consequently, PAG and PAO do not achieve a meaningful improvement over strong naive baselines such as IFP.

In the random marginal benefit setting, early commitments to treating marginals can become costly. For example, treating an individual to an intermediate level may be dominated by initiating a full treatment on a different individual during a subsequent budget level. This benefit structure is precisely the regime where the nested (prefix-based) constraint becomes binding. PAG can perform well at early budgets by selecting the best feasible next step greedily, but its requirement that allocations remain nested limits its ability to revise earlier choices as the budget grows. By recomputing the allocation at each budget, PAO can reallocate away from partially treated individuals when a better frontier emerges, which explains its consistent pointwise dominance across budgets.

In the strictly decreasing marginal benefit setting, most of the valuable treatment marginals are able to be treated first, and therefore the best corresponding policy exhausts earlier treat-

ment levels across many individuals. Because later levels are consistently less valuable than earlier levels, there is little benefit to reallocation. This explains why PAO offers no systematic advantage over PAG in this regime, even though it can still outperform IFP due to IFP's commitment to fully treating individuals.

Overall, these findings clarify when multi-level allocation meaningfully departs from the binary treatment setting. In binary allocation, policies naturally correspond their ranking output, so a policy to allocate treatment effects further, such as a policy in our case, is unnecessary. In the multi-level setting, this equivalence can break down. When marginal benefits vary across treatment levels, maximizing realized treatment effect may require policies that change which individuals they prioritize as the budget evolves. In such cases, allocation cannot be represented by a single nested ranking, and flexible policies such as PAO provide a structural advantage.

6.4 Non-Nested Allocations In The Clinical Setting

A potential limitation of non-nested allocation policies such as PAO is their lack of interpretability and feasibility in real-world clinical settings. For example, in a setting where a hospital has already allocated treatment up to its current budget, the arrival of an additional unit of budget introduces ambiguity in how that resource should be deployed. Unlike nested policies, which extend naturally by following a fixed ranking, PAO may require re-solving the allocation problem entirely, potentially revising earlier decisions. This raises a real question of whether an incremental increase should trigger a reallocation or whether they should remain stable once allocated.

In practice, we propose that PAO should not be implemented as a full dynamic reallocation at each budget increment. Rather, allocation decisions can be made at regular intervals, such as shift changes in hospitals, when resources are known or fixed. This avoids the need to re-optimize allocations in response to small, incremental changes in budget, where solving

a full allocation problem to assign a single additional treatment may be impractical.

We encourage this problem to be explored further in future work, particularly in designing policies that balance allocation flexibility with the need for stability and interpretability in clinical environments.

6.5 On The Use Of Non-Synthetic Data

A natural question that arises at the end of this research is whether the findings in the paper can be extended to real-world data. While evaluating allocation policies on observational data is highly desirable, the implementation of the multi-level treatment setting on non-synthetic data is difficult.

First, the evaluation of allocation policies in our framework requires access to individual-level *marginal treatment effects* for each treatment level. In real-world datasets, these quantities are inherently unobserved. Estimating these quantities requires strong assumptions of consistency, unconfoundeness, and overlap across all treatment levels [19]. In multi-level settings, the overlap assumption becomes particularly restrictive, as each individual must have a non-zero probability of receiving every treatment intensity.

Even when these assumptions hold, estimating marginal treatment effects at multiple levels introduces statistical noise. Because the multi-level setting requires estimating a sequence of incremental effects, the errors in the compounding of these estimates make it difficult to disentangle where policy performance comes from.

That said, extending this work to non-synthetic data is important future work. One promising approach could be the use of semi-synthetic data where real covariates and treatment assignments are combined with simulated outcome models. More broadly, applying the ML-AUTOC framework with high-quality causal estimators and explicitly accounting for estimation uncertainty will be critical for understanding how these allocation policies perform in practice.

Chapter 7

Conclusion

This thesis investigates whether precedence-aware greedy and reallocation policies outperform naive ranking-based policies in terms of achieving higher realized total average treatment effect in multi-level treatment settings. Our findings show that reallocation for optimal benefit (PAO) is not necessary to achieve strong performance. In settings with strictly ordered marginal benefits, PAG achieves performance comparable to naive ranking-based policies.

However, in settings where there are random marginal benefits for different treatment levels, the relaxation of the nesting constraint in PAO is shown to have higher pointwise performance across different budgets, suggesting that in these settings, it is important to maximize the realized treatment effect by reallocating resources rather than relying on a naive approach.

Bibliography

[1] Efren P. Agonafer, Susan L. Carson, Victoria Nunez, Fiona Jones, Lori Carter-Edwards, Oluwasegun Adekeye, Jessica Resendez, Marsha R. Weintraub, J. Nwando Olayiwola, and Arleen F. Brown. Community-based organizations' perspectives on improving health and social service integration. *BMC Public Health*, 2021. doi: 10.1186/s12889-021-10449-w. URL `https://doi.org/10.1186/s12889-021-10449-w`.

[2] Ayse Aslan, Evrim Ursavas, and Ward Romeijnders. A precedence constrained knapsack problem with uncertain item weights for personalized learning systems. *Omega*, 115: 102779, 2023. ISSN 0305-0483. doi: https://doi.org/10.1016/j.omega.2022.102779. URL `https://www.sciencedirect.com/science/article/pii/S0305048322001864`.

[3] Richard Bellman. *Dynamic Programming*. Dover Publications, 1957. ISBN 9780486428093.

[4] Dimitris Bertsimas and John N. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, 1997.

[5] Elizabeth C. Caniglia, Erica E. M. Moodie, Roger W. Logan, and Miguel A. Hernán. Estimating optimal dynamic treatment strategies under resource constraints using dynamic marginal structural models. *Statistics in Medicine*, 2021. doi: 10.1002/sim.9177. URL `https://pmc.ncbi.nlm.nih.gov/articles/PMC9017598/`.

[6] Michael F. Drummond, Mark J. Sculpher, Karl Claxton, Greg L. Stoddart, and

George W. Torrance. *Methods for the Economic Evaluation of Health Care Programmes*. Oxford University Press, 2015.

[7] G. French, M. Hulse, D. Nguyen, K. Sobotka, B. Webster, J. Corman, B. Aboagye-Nyame, M. McNulty, K. Klinker, G. Dumyati, and et al. Impact of hospital strain on excess deaths during the covid-19 pandemic — united states, july 2020–july 2021. *MMWR. Morbidity and Mortality Weekly Report*, 2021. URL https://www.cdc.gov/mmwr/volumes/70/wr/mm7046a5.htm.

[8] Arnaud Fréville. The multiple-choice knapsack problem: An overview. *European Journal of Operational Research*, 113(1):1–14, 2004.

[9] Nicole B. Gabler, Sarah J Ratcliffe, Jason Wagner, David A. Asch, Gordon D. Rubenfeld, Derek C. Angus, and Scott D. Halpern. Mortality among patients admitted to strained intensive care units. *Am J Respir Crit Care Med*, 2013. URL https://pmc.ncbi.nlm.nih.gov/articles/PMC3826272/.

[10] T. C. Hu. Parallel sequencing and assembly line problems. *Operations Research*, 9 (6):841–848, 1961. ISSN 0030364X, 15265463. URL http://www.jstor.org/stable/167050.

[11] Fahad Kamran*, Shengpu* Tang, Erkin Otles, Dustin S McEvoy, Sameh N Saleh, Jen Gong, Benjamin Y Li, Sayon Dutta, Xinran Liu, Richard J Medford, Thomas S Valley, Lauren R West, Karandeep Singh, Seth Blumberg, John P Donnelly, Erica S Shenoy, John Z Ayanian, Brahmajee K Nallamothu, Michael W Sjoding†, and Jenna Wiens†. Early identification of patients admitted to hospital for covid-19 at risk of clinical deterioration: model development and multisite external validation study. *The British Medical Journal (BMJ)*, 2022. doi: 10.1136/bmj-2021-068576.

[12] Fahad Kamran, Maggie Makar, and Jenna Wiens. Learning to rank for optimal treatment allocation under resource constraints. In *Proceedings of The 27th International*

Conference on Artificial Intelligence and Statistics, pages 1–18. PMLR, 2024. URL https://proceedings.mlr.press/v238/kamran24a/kamran24a.pdf.

[13] Hans Kellerer, Ulrich Pferschy, and David Pisinger. *Knapsack Problems*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004. ISBN 3-540-40286-1, 978-3-540-40286-2. doi: 10.1007/978-3-540-24777-7.

[14] Donald Knuth. *The Art Of Computer Programming, vol. 3: Sorting And Searching*. Addison-Wesley, 1973.

[15] Sören R. Künzel, Jasjeet S. Sekhon, Peter J. Bickel, and Bin Yu. Metalearners for estimating heterogeneous treatment effects using machine learning. *Proceedings of the National Academy of Sciences*, 116(10):4156–4165, 2019. doi: 10.1073/pnas.1804597116. URL https://www.pnas.org/doi/10.1073/pnas.1804597116.

[16] Silvano Martello and Paolo Toth. *Knapsack Problems: Algorithms and Computer Implementations*. Wiley, 1990.

[17] Eduardo Moreno, Daniel Espinoza, and Marcos Goycoolea. Large-scale multi-period precedence constrained knapsack problem: A mining application. *Electronic Notes in Discrete Mathematics*, 36:407–414, 08 2010. doi: 10.1016/j.endm.2010.05.052.

[18] David Pisinger. Where are the hard knapsack problems? *Computers & Operations Research*, 32(9):2271–2284, 2005.

[19] Donald B. Rubin. Estimating causal effects of treatments in randomized and non-randomized studies. *Journal of Educational Psychology*, 66(5):688–701, 1974. doi: 10.1037/h0037350. URL https://doi.org/10.1037/h0037350.

[20] Mehran Samavati, Daryl Essam, Micah Nehring, and Ruhul Sarker. A methodology for the large-scale multi-period precedence-constrained knapsack problem: an application in the mining industry. *International Journal of Production Economics*, 193:12–20,

2017. ISSN 0925-5273. doi: https://doi.org/10.1016/j.ijpe.2017.06.025. URL https://www.sciencedirect.com/science/article/pii/S0925527317301986.

[21] Uri Shalit, Fredrik D Johansson, and David Sontag. Estimating individual treatment effect: generalization bounds and algorithms. In *International conference on machine learning*, pages 3076–3085. PMLR, 2017.

[22] Rahim Vakili, Saba Vakili, Maryam Ajilian Abbasi, Shokofeh Masoudi, and Masumeh Saeidi. Overcrowded classrooms: Challenges, consequences, and collaborative solutions for educators: A literature review. *Medical Education Bulletin*, 2024. URL http://medicaleducation-bulletin.ir/article_210691.html.

[23] Steve Yadlowsky, Scott Fleming, Nigam Shah, Emma Brunskill, and Stefan Wager. Evaluating treatment prioritization rules via rank-weighted average treatment effects. *arXiv preprint arXiv:2110.15345*, 2021.

[24] Lei Zhao, Lu Tian, Tianxi Cai, Brian Claggett, and Lee-Jen Wei. Effectively selecting a target population for a future comparative study. *Journal of the American Statistical Association*, 108(502):527–539, 2013. URL https://doi.org/10.1080/01621459.2013.770703.

Appendix A

Appendix

A.1 Heterogeneous Treatment Costs

The primary analysis in this thesis assumes that each marginal treatment level incurs a unit cost. While this simplifies the allocation problem and isolates the role of marginal benefit structure, it abstracts away an important feature of many real-world intervention systems: treatment intensity often varies in cost.

In practice, higher levels of intervention typically require greater resource expenditure. For example, escalating clinical treatment intensity may require additional staff time, specialized services, or longer duration of care [6]. Cost heterogeneity is therefore not merely a modeling convenience but reflects structural properties of real-world resource allocation problems.

Allowing treatment levels to differ in cost aligns the framework more closely with standard welfare maximization and cost-effectiveness formulations in economics and operations research, where decision-makers face a fixed budget and must allocate heterogeneous interventions to maximize total benefit [4].

To start, let $c_\ell > 0$ denote the marginal cost of advancing an individual from treatment level $\ell - 1$ to level ℓ. We define the cumulative cost of assigning individual i to level ℓ as

$$C(\ell) = \sum_{k=1}^{\ell} c_k,$$

which is weakly increasing in ℓ.

Given marginal benefits $\delta_i^{(\ell)}$ as defined in 2.1, the allocation problem under a fixed total budget B becomes:

$$\max_{\{z_{i,\ell}\}} \sum_{i=1}^{n} \sum_{\ell=1}^{L} \delta_i^{(\ell)} z_{i,\ell}$$

subject to

$$\sum_{i=1}^{n} \sum_{\ell=1}^{L} c_\ell z_{i,\ell} \leq B,$$

$$z_{i,\ell} \leq z_{i,\ell-1} \quad \forall i, \ell \geq 2,$$

$$z_{i,\ell} \in \{0, 1\}.$$

The second constraint enforces precedence: an individual may only receive level ℓ if all prior levels have been assigned. The problem now resembles a precedence-constrained knapsack problem, where each marginal treatment increment yields benefit $\delta_i^{(\ell)}$ but consumes c_ℓ units of budget [18]. The individual optimizing must therefore trade off both benefit magnitude and cost-effectiveness.

Under unit costs, the budget parameter β directly corresponds to the number of marginal treatment increments allocated. Each step in the ML-TOC curve therefore reflects one additional unit of treatment.

With heterogeneous costs, this interpretation changes. The budget B now measures total expenditure rather than number of treatment increments. A single additional allocation may consume multiple units of budget, and different increments may advance the budget frontier by unequal amounts.

Consequently:

1. The horizontal axis of ML-TOC curves would represent cumulative expenditure rather than count of treatment steps.

2. Budget increments become non-uniform.

3. The mapping between allocation rank and budget becomes nonlinear.

We believe that introducing heterogeneous costs should increase the importance of flexible, non-nested allocation policies. Under unit costs, the primary trade-off concerns only marginal benefit magnitude. However, when costs differ, optimal allocation depends on benefit-to-cost ratios:

$$\frac{\delta_i^{(\ell)}}{c_\ell}.$$

If higher treatment levels have disproportionately high costs, it may become optimal to partially treat multiple individuals rather than fully treat a small number. Conversely, if early levels are inexpensive but later levels are costly, the optimal policy may exhibit switching behavior across individuals to maximize aggregate benefit per dollar spent.

We therefore hypothesize:

> *When treatment costs are heterogeneous, precedence-aware policies that allow reallocation (e.g.,* PAO*) will outperform ranking-based or nested greedy policies by a larger margin than in the unit-cost setting.*

In particular, heterogeneous costs introduce an additional dimension along which early allocation decisions may become suboptimal at later budget levels, increasing the value of dynamic reoptimization.

A.2 PGP, IFP, LFP, and PAG Code Implementation

PGP Code

```
# Pooled Global Policy (PGP)

initialize empty list of pairs

# treat every (i, l) as a separate item
for each individual i:
    for each level l:
        add (i, l) to list

# sort all pairs by marginal benefit
sort list of pairs by delta[i][l] in descending order

ranking = sorted list
return ranking
```

IFP Code

```
# Individual-First Policy (IFP)

initialize empty list ranking

# compute total value for each individual
for each individual i:
    total[i] = sum over l of delta[i][l]

# sort individuals by total value
```

```
    create list of individuals sorted by total[i] in descending order

    # assign all levels for each individual before moving on
    for each individual i in sorted list:
        for each level l from 1 to L:
            append (i, l) to ranking

    return ranking
```

LFP Code

```
    # Level-First Policy (LFP)

    initialize empty list ranking

    for each level l from 1 to L:

        # sort individuals by marginal benefit at this level
        create list of individuals sorted by delta[i][l] in descending order

        for each individual i in this sorted list:
            append (i, l) to ranking

    return ranking
```

PAG Code

```
    # Precedence-Aware Greedy (PAG)

    initialize empty ranking list
```

```
# priority queue stores (i, l) with priority delta[i][l]
initialize empty max-priority queue

# initially, only level 1 is feasible for each individual
for each individual i:
    push (i, 1) into priority queue with priority delta[i][1]

# iterate over full budget
for b = 1 to nL:
    # select best feasible marginal
    (i, l) = pop max from priority queue

    add (i, l) to ranking

    # unlock next level for that individual
    if l + 1 <= L:
        push (i, l+1) into priority queue
        with priority delta[i][l+1]

return ranking
```

A.3 Dynamic Programming for PAO Implementation

Dynamic programming was used to implement the PAO policy. The formulation is closely related to classical knapsack-style dynamic programming, in which a finite resource budget is allocated across multiple items to maximize total value [3, 13].

In this setting, each individual corresponds to a group of mutually exclusive choices indexed by $t_i \in \{0, 1, \ldots, L\}$, and the total budget is nL [13]. Dynamic programming provides an exact solution by tracking the optimal achievable benefit for each intermediate budget level. In order to use this approach, prefix sums of marginal benefits are computed, which allows for a constant-time computation of the assignment of treatment levels to a particular individual. The dynamic program iterates through individuals and prefix lengths, updating the value function on all budgets. The transition of the value function is shown in the following code:

```
# PAO: Dynamic programming over individuals and budgets

n = number of individuals
L = number of treatment levels
B = n * L

# Step 1: compute prefix values
# prefix[i][m] = total value of assigning the first m levels to individual i
initialize prefix as an n x (L + 1) table of zeros

for each individual i:
    prefix[i][0] = 0
    for each level m from 1 to L:
        prefix[i][m] = prefix[i][m - 1] + delta[i][m]

# Step 2: initialize dynamic program
# dp_prev[b] = best total value achievable using processed
# individuals and budget b
initialize dp_prev[0] = 0
initialize dp_prev[b] = -infinity for all b > 0
```

```
# choice[i][b] will store how many levels were assigned to individual i
# in the optimal solution at budget b during the i-th step
initialize choice as an n x (B + 1) table of zeros

# Step 3: process individuals one at a time
for each individual i:

    # start with the option of assigning no treatment to this individual
    best[b] = dp_prev[b] for every budget b
    best_m[b] = 0 for every budget b

    # consider assigning a prefix of length m to individual i
    for each prefix length m from 1 to L:

        # candidate value if m treatment units are assigned to individual i
        for each budget b from m to B:
            cand[b] = dp_prev[b - m] + prefix[i][m]

            # if this improves the best known value at budget b, update it
            if cand[b] > best[b]:
                best[b] = cand[b]
                best_m[b] = m

    # store the chosen prefix length for this individual at each budget
    for each budget b from 0 to B:
        choice[i][b] = best_m[b]
```

```
    # move to the next individual
    dp_prev = best

# Step 4: recover allocations by backward pass
# For each budget beta, trace backward through choice to determine
# how many levels were assigned to each individual
for each budget beta from 1 to B:
    remaining_budget = beta

    for individuals i in reverse order:
        m = choice[i][remaining_budget]
        assign levels 1 through m to individual i
        remaining_budget = remaining_budget - m
```

To recover feasible allocations, the selected prefix length for each individual and budget is stored during the forward pass and used in a backward pass to reconstruct the allocation corresponding to each budget level.

The full implementation details, including the dynamic programming routine and dataset generation procedures, are available online at `https://mlautoc.com`.

A.4 Dataset Generation

Dataset A: Strictly Increasing Marginal Benefits

$$x_i \sim \mathcal{N}(0, I_d)$$

$$z_i \sim \mathcal{N}(\mu \mathbf{1}_L,\ 0.5^2 I_L)$$

$$\delta_i^{(1)} = \max\{0,\ z_i^{(1)}\}$$

$$\delta_i^{(\ell)} = \max\left\{z_i^{(\ell)},\ \delta_i^{(\ell-1)} + 1\right\} \text{ for } \ell = 2, \ldots, L$$

Dataset B: Random Marginal Benefits

$$x_i \sim \mathcal{N}(0, I_d)$$

$$z_i \sim \mathcal{N}(\mu \mathbf{1}_L,\ 0.5^2 I_L)$$

$$\delta_i^{(\ell)} = \max\{0,\ z_i^{(\ell)}\} \text{ for } \ell = 1, \ldots, L$$

Dataset C: Strictly Decreasing Marginal Benefits

Dataset C is constructed as the mirror image of Dataset A. For each individual i, the strictly increasing marginal benefit sequence $\{\delta_i^{(1)}, \ldots, \delta_i^{(L)}\}$ generated under Dataset A is reversed to obtain $\{\delta_i^{(L)}, \ldots, \delta_i^{(1)}\}$, yielding strictly decreasing marginal benefits across treatment levels.

Dataset D: Heterogeneous Strictly Increasing Marginal Benefits

$$s_i \sim \text{LogNormal}(0,\ \sigma_s^2)$$

$$z_i^{(\ell)} \sim \mathcal{N}(s_i,\ 0.5^2)$$

$$\delta_i^{(1)} = \max\{0,\ z_i^{(1)}\}$$

$$\delta_i^{(\ell)} = \max\left\{z_i^{(\ell)},\ \delta_i^{(\ell-1)} + 1\right\} \text{ for } \ell = 2, \ldots, L$$

Each individual i is assigned a scale s_i, which determines how large their treatment effects are overall. The values $z_i^{(\ell)}$ are noisy draws of the marginal benefit at each level, centered around s_i.

The final marginal benefits $\delta_i^{(\ell)}$ are constructed from these draws. The first level is truncated to be nonnegative, and each subsequent level is forced to be at least one unit larger than the previous level. This ensures that marginal benefits strictly increase for each individual, while allowing different individuals to have very different magnitudes of treatment effects.

Appendix B

B.1 Run Time Data

Table B.1: Runtime (seconds) of ranking policies as a function of dataset size n on strictly increasing marginal benefit datasets with $L = 4$.

n	**PGP**	**IFP**	**LFP**	**PAO**	**PAG**
500	0.0008	0.0009	0.0009	0.6721	0.0041
1,000	0.0017	0.0017	0.0017	2.6607	0.0086
2,000	0.0033	0.0033	0.0033	10.7443	0.0178
4,000	0.0069	0.0069	0.0069	43.1270	0.0483
8,000	0.0143	0.0135	0.0135	176.8498	0.1529

B.2 Software

To enable the repeatability and efficiency in the experimental phase of this research, we have developed an accompanying software that allows for ML-AUTOC demonstrations available at **mlautoc.com**. As stated in the docs, the goal of the software was to facilitate the following:

1. Choose the number of patients in the dataset.

2. Select the number of marginal treatment levels allocated to each patient has.

3. Experiment with the cost structure of the experiment.

4. Configure the underlying structure of marginal treatment effects.

5. Optionally upload an individual dataset (CSV) and run an experiment.

6. View resource allocation graphs and ML-AUTOC scores.

7. Analyze runtime performance (time logged and displayed as a graph).

The platform is implemented using **Next.js**, and the backend runs locally on the experimenter's computer. Experiments are executed using common Python libraries to ensure full reproducibility. Source code and documentation are available on the project website under **mlautoc.com/experiments**.

During research we developed several functions that allows us to generate, rank, and visualize controlled treatment allocation scenarios prior to web deployment. These utilities were used to test our theories on flip rankings, precedence enforcement, and cost heterogeneity before web implementation. These functions can be downloaded as files under **mlautoc.com/downloads**.

If the website is inaccessible, all available code can be found on GitHub at: **https://github.com/gilstero/ML-TOC**.