ICS NEWS
INFORMS Computing Society Newsletter

Inside This Issue...

"The Society is a great opportunity to get involved, meet some new people and generally combine your professional interests with a little fun."

Message from the Chair

Merve Bodur
School of Mathematics, The University of Edinburgh
merve.bodur@ed.ac.uk

Dear ICS colleagues, I am delighted to write my first newsletter message as Chair of the INFORMS Computing Society. Having served as Vice-Chair for the past two years, I have seen more closely how much of ICS is sustained by its members: the colleagues who give talks, organize and chair sessions, serve on award committees, help shape our meetings and conferences, mentor students, contribute to our newsletters, and keep our society connected across meetings and across years.

I would like to begin by thanking Thiago Serra for his leadership as Chair during 2024-2025, and Kai Pan for his service as Secretary/Treasurer. I also thank Harsha Gangammanavar for his service on the ICS Board. I am very happy to welcome Andre A. Cire as Vice-Chair/Chair-Elect, Jose L. Walteros as Secretary/Treasurer, and Thiago Serra and Allen Holder as new Board members. Beste Basciftci, Austin Buchanan, and Ricardo Fukasawa continue their Board service, and Hamed Rahimian continues as our Newsletter Editor and Webmaster. We are fortunate to have such a strong group helping guide ICS.

The past year gave us much to build on. The 2025 ICS Conference in Toronto brought our community together in a lively and successful meeting; many thanks to Andre Cire and Sheng Liu for leading that effort. Our awards program continues to recognize excellent work at the interface of computing and operations research, and our membership has grown to exceed 1,000 members. That number is encouraging, but what matters more is what it represents: students, scholars, and practitioners who see ICS as a natural home for computational ideas in OR.

I also want to recognize the continuing importance of the INFORMS Journal on Computing to our society. Many thanks to Alice Smith for her remarkable service as Editor in Chief, and a warm welcome to Andrea Lodi as the new Editor in Chief. The journal's growth, its attention to software and data, and its evolving coverage of emerging computational areas speak directly to the role ICS can play in shaping research practice.

Looking ahead, our next opportunity to meet will be at the 2026 INFORMS Annual Meeting in San Francisco. I hope many of you will join the ICS sessions, attend the business meeting, and use the meeting to reconnect with colleagues and welcome new members to the society. After that, our attention will turn to the next ICS Conference. Planning for ICS 2027 is already underway. Please mark your calendars for March 8-10, 2027, when the ICS Conference will be held at Carnegie Mellon University in Pittsburgh! More details will follow soon.

At its best, ICS is where strong technical ideas meet a generous professional community. I hope you will stay involved: nominate colleagues and students for awards, organize sessions, share news for future newsletters, and consider serving the society. I look forward to seeing many of you in San Francisco and continuing the conversation there.

Message from the Newsletter Editor

Hamed Rahimian
Department of Industrial Engineering, Clemson University
hrahimi@clemson.edu

As we present this edition of ICS News, I am delighted to share research highlights from the 2025 ICS award recipients. I am especially pleased that seven of the ten awardees contributed articles showcasing the innovative work recognized by these awards. I encourage you to explore this issue and engage with the scholarship that makes our professional community so vibrant. As always, I welcome your feedback. All past issues of ICS News are available at https://connect.informs.org/computing/ics-resources/newsletters. You can also follow us on Bluesky at @icsinforms.

Message from the Editor-in-Chief of INFORMS Journal on Computing

Andrea Lodi
Jacobs Technion-Cornell Institute, Cornell Tech and Technion - IIT
andrea.lodi@cornell.edu

I am very happy to address my first message to the ICS community through this newsletter. As the new Editor in Chief of INFORMS Journal on Computing (IJOC), it is very important to me to establish an enthusiastic relationship with this community and to ensure that IJOC is your journal of choice for your most exciting work.
As many of you have already read in my first editorial in March, the editorial team has been significantly renewed. I am grateful to Shane Henderson, Willem-Jan van Hoeve, Simge Küçükyavuz, Jim Luedtke, Ruth Misener, Veronica Piccialli, and Domenico Salvagnin for enthusiastically accepting new roles as Area Editors, and to Russell Bent and Giacomo Nannicini for remaining on board and providing valuable continuity.
We have also introduced several structural changes to the editorial areas. These include new area titles, such as "Artificial Intelligence & Optimization," as well as the creation of two new areas—"Computational Modeling & Applications" and "Simulation, Stochastic Models, & Stochastic Optimization"—each formed by consolidating previously existing areas. Let me also reassure the community that, although biological systems, medicine, and healthcare no longer constitute a separate editorial area, IJOC remains strongly committed to publishing high-quality research in these domains.
These changes are intended to reflect the evolution of our community, including the increasing role of AI and the growing importance of code and data availability to support reproducibility. These developments present both challenges and opportunities. Our ambition is for IJOC to remain the premier venue for impactful research in computing, where methodological innovation and practical relevance go hand in hand. As editors, we greatly value feedback from the ICS community, so please do not hesitate to reach out to us at any time.
Finally, let me take this opportunity to thank the former Editor in Chief, Alice Smith, and the entire previous editorial team for leaving IJOC in such a healthy state. Their outstanding work has provided an excellent foundation on which we will continue to build.

Harvey J. Greenberg Research Award

The award honors research excellence in the field of computation and operations research applications, especially those in emerging application fields. Honored research would focus on contributions that exhibit the promise of making a significant impact in the scope of OR/MS/Analytics practice.

Winners: Braden L. Crimmins, J. Alex Halderman, and Bradley Sturt for their paper "Improving the Security of United States Elections with Robust Optimization."

Honorable Mention: Soroush Saghafian for his paper "Ambiguous Dynamic Treatment Regimes: A Reinforcement Learning Approach."

Honorable Mention: Xiao-Yue Gong and David Simchi-Levi for their paper "Bandits atop Reinforcement Learning: Tackling Online Inventory Models with Cyclic Demands."

Chairs: Selva Nadarajah and Andre A. Cire.

Steering Committee: Dan Adelman; David Brown; Ricardo Fukasawa; Simge Küçükyavuz; Siqian Shen; and Golbon Zakeri.

Review Panel: Yi-Chun Akchen; Margarida Carvalho: Margarita Paz Castro: Levi DeValve: Ludwig Dierks; Daniel Jiang; Carla Michini; and Raghav Singal.

ICS Prize

The ICS Prize is an annual award for the best English language paper or group of related papers dealing with the Operations Research/Computer Science interface. Due to the exceptional submissions received this year, the committee recommended the recognition of two submissions as co-winners of the ICS Prize, as well as one submission as the recipient of Honorable Mention.

Co-Winners: Alper Atamtürk, Andrés Gómez, and Shaoning Han for their pioneering contributions to modeling and solving mixed-integer quadratic optimization problems.

Co-Winners: Jason Altschuler and Pablo Parrilo for their pioneering work on accelerating gradient descent through stepsize hedging.

Honorable Mention: Shabbir Ahmed (in memoriam), Yongpei Guan, Ruiwei Jiang, and Weijun Xie for their fundamental contributions to the computation of distributionally robust chance-constrained programs (DRCCPs).

Winning materials

Co-Winner: Alper Atamtürk, Andrés Gómez, and Shaoning Han

- Alper Atamtürk and Andrés Gómez. Strong formulations for quadratic optimization with M-matrices and indicator variables. *Mathematical Programming, Series B* (2018) 170:141–176.
- Alper Atamtürk and Andrés Gómez. Submodularity in Conic Quadratic Mixed 0–1 Optimization. *Operations Research*, Vol. 68, No. 2, March–April 2020, pp. 609–630.
- Alper Atamtürk and Andrés Gómez. Safe Screening Rules for ℓ_0-Regression from Perspective Relaxations. Proceedings of the 37th International Conference on Machine Learning, PMLR 119, 2020.
- Alper Atamtürk, Andrés Gómez, and Shaoning Han. Sparse and Smooth Signal Estimation: Convexification of ℓ_0-Formulations. *Journal of Machine Learning Research* 22 (2021) 1–43.
- Shaoning Han, Andrés Gómez, and Alper Atamtürk. 2×2-Convexifications for convex quadratic optimization with indicator variables. *Mathematical Programming* (2023) 202:95–134.
- Alper Atamtürk and Andrés Gómez. Rank-one Convexification for Sparse Regression. *Journal of Machine Learning Research* (2025) 1–50.

Co-Winner: Jason Altschuler and Pablo Parrilo

- Jason M. Altschuler and Pablo A. Parrilo. 2025. Acceleration by Stepsize Hedging: Multi-Step Descent and the Silver Stepsize Schedule. *J. ACM* 72, 2, Article 12 (March 2025), 38 pages. doi.org/10.1145/3708502.
- Jason M. Altschuler and Pablo A. Parrilo. Acceleration by stepsize hedging: Silver Stepsize Schedule for smooth convex optimization. *Mathematical Programming*, doi.org/10.1007/s10107-024-02164-2, 2024.

Honorable Mention: Shabbir Ahmed (in memoriam), Yongpei Guan, Ruiwei Jiang, and Weijun Xie

- Ruiwei Jiang and Yongpei Guan. Data-driven chance constrained stochastic program. *Mathematical Programming, Series A* (2016) 158:291–327. 10.1007/s10107-015-0929-7.
- Weijun Xie, Shabbir Ahmed, and Ruiwei Jiang. Optimized Bonferroni approximations of distributionally robust joint chance constraints. *Mathematical Programming* (2022) 191:79–112. doi.org/10.1007/s10107-019-01442-8.
- Weijun Xie. On distributionally robust chance constrained programs with Wasserstein distance. *Mathematical Programming* (2021) 186:115–155. doi.org/10.1007/s10107-019-01445-5.
- Weijun Xie and Shabbir Ahmed. Bicriteria Approximation of Chance-Constrained Covering Problems. *Operations Research*, Vol. 68, No. 2, March–April 2020, pp. 516–533.

Committee members: George Lan (Chair); Güzin Bayraksan; Grani Hanasusanto; and Andrew Trapp.

ICS Best Student Paper Award

The ICS Student Paper Award is an annual award for the best paper at the interface of computing and operations research by a student author. **The 2025 winner was awarded to Hao Hao (CMU)** for the paper "Robust Paths: Geometry and Computation," co-authored with Peter Zhang.

Runner-up: Thomas Hübner (ETH Zürich) for the paper "Spatial branch-and-bound for nonconvex separable piecewise linear optimization," co-authored with Akshay Gupte and Steffen Rebennack.

Honorable Mention: Yupeng Wu (London Business School) for the paper "The Surprising Performance of Random Partial Benders Decomposition," co-authored by Jean Pauphilet.

Honorable Mention: Matías Villagra (Columbia) for the paper "Accurate Linear Cutting-Plane Relaxations for ACOPF," co-authored by Daniel Bienstock.

Committee members: Ryan Cory-Wright; Austin Buchanan (Chair); Yongchun Li; and Young Woong Park.

Research Highlight
Improving the Security of United States Elections with Robust Optimization

by Braden L. Crimmins[1], J. Alex Halderman[1], Bradley Sturt[2]
[1]Computer Science and Engineering, University of Michigan, Ann Arbor
[2]Information and Decision Sciences, University of Illinois Chicago

Introduction: The Vulnerabilities of Current Logic and Accuracy Testing

Computerized voting machines are essential for determining election outcomes across the United States. Voters often participate in a large number of contests—from the President down to local school boards—that make hand counting impractically slow and costly. For these machines to count votes accurately, they must be configured with the correct mapping between the voting targets on a ballot and the corresponding candidates.

If this mapping is misconfigured, whether accidentally or maliciously, votes can be swapped across candidates, producing dramatically wrong totals. Recent elections have seen accidental misconfigurations in Antrim County, Michigan [6], as well as in Pennsylvania [3] and Georgia [5], which generated significant negative publicity and damaged public trust. Furthermore, adversaries could strategically induce these misconfigurations to manipulate outcomes with very little technical expertise [4].

For over a century, election officials have inspected voting machines before elections using a procedure called Logic and Accuracy Testing (LAT) (see Figure 1). Officials cast a “test deck" of filled-out ballots into the machine to confirm it produces the expected vote totals. However, test decks today are designed using simple human heuristics, such as assigning one vote to the first candidate, two to the second, and so on [7]. These heuristics are mathematically blind to specific types of misconfigurations—meaning a compromised

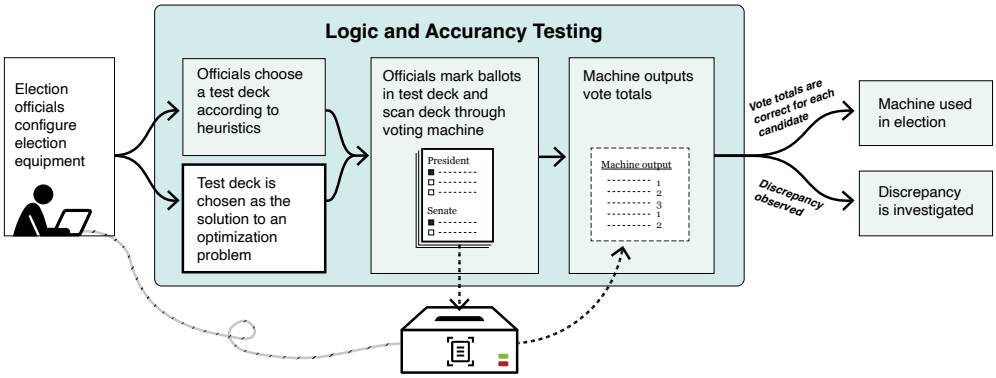


Figure 1: Visualization of Logic and Accuracy Testing (LAT). The proposed mathematical modification ensures LAT catches sophisticated misconfigurations.

machine could successfully pass LAT while still swapping votes in the actual election.

A Scientific Approach: Robust Logic and Accuracy Testing (RLAT)

To bring a scientific perspective to LAT, we introduce the first formal approach to designing test decks with rigorous security guarantees. By employing tools from the field of robust optimization [1, 2], we developed a framework called Robust Logic and Accuracy Testing (RLAT).

Unlike heuristic-based decks, RLAT uses mathematical optimization to find a test deck that is guaranteed to detect any misconfiguration that swaps votes between candidates. In our framework, we consider a “ballot style" consisting of a set of contests $\mathcal{C}$ and a set of candidates $\mathcal{N}$. A test deck is a finite-length sequence of filled-out ballots $(\beta_1, \ldots, \beta_B)$.

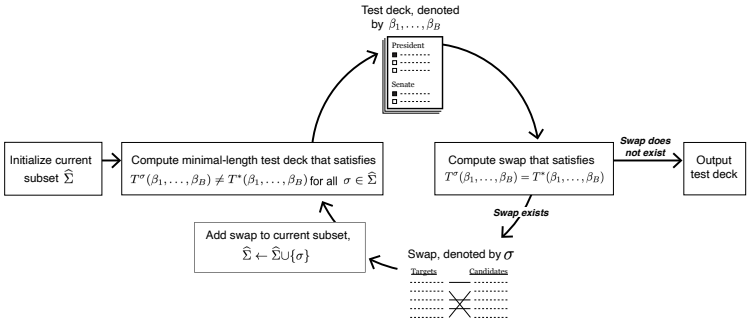


Figure 2: Visualization of the exact algorithm solving the robust optimization problem.

When a voting machine operates correctly, its output is represented by the vector-valued function $T^*(\beta_1, \ldots, \beta_B)$. If a machine is operating incorrectly, its output is represented by $\widehat{T}(\beta_1, \ldots, \beta_B)$. Given an uncertainty set $\mathcal{U}$ of possible misconfigurations, RLAT designs the test deck by solving the following general robust optimization problem (RO):

$$\begin{aligned} \underset{B \in \mathbb{N},\ \beta_1, \ldots, \beta_B \in \mathscr{B}}{\text{minimize}} \quad & B \\ \text{subject to} \quad & \widehat{T}(\beta_1, \ldots, \beta_B) \neq T^*(\beta_1, \ldots, \beta_B) \quad \forall \widehat{T}(\cdot) \in \mathcal{U}. \end{aligned} \tag{RO}$$

This optimization problem yields a minimum-length test deck (B) that is guaranteed to detect whether a voting machine is operating incorrectly in any of the ways specified by the uncertainty set.

Defining the Swap Uncertainty Set

To provide strict security against vote-swapping, we define a specific "swap uncertainty set" Σ. This set consists of all possible voting machines that have an incorrect bijective mapping from candidates to voting targets. For a non-identity bijection $\sigma \in \Sigma$, the machine's output is $T^\sigma(\beta_1, \ldots, \beta_B)$.

Equipped with this set, we narrow our general model to the specific problem (RO-Σ):

$$\begin{aligned} \underset{B \in \mathbb{N},\ \beta_1, \ldots, \beta_B \in \mathscr{B}}{\text{minimize}} \quad & B \\ \text{subject to} \quad & T^\sigma(\beta_1, \ldots, \beta_B) \neq T^*(\beta_1, \ldots, \beta_B) \quad \forall \sigma \in \Sigma. \end{aligned} \tag{RO-Σ}$$

Out of all the test decks that carry this strict security guarantee, our optimization problem identifies the deck with the absolute minimum number of ballots, minimizing the operational cost and time burden for election officials.

Overcoming the Computational Challenge

While the RLAT framework provides a rigorous security guarantee, finding the optimal test deck is computationally challenging. In a ballot style with N candidates, there are $|\Sigma| = N! - 1$ possible ways that voting targets can be swapped. Because real-world ballot styles often feature over 100 candidates, the optimization problem must account for more than $100! - 1 \approx 10^{157}$ constraints. This makes it impossible to solve on any extant computer using standard methods.

To circumvent the need to solve an optimization problem with a virtually infinite number of constraints, we developed an exact algorithm inspired by the cutting plane method (see Figure 2). Rather than explicitly coding every possible swap, the algorithm solves a relaxed version of the problem containing only a small subset of possible swaps, $\widehat{\Sigma} \subseteq \Sigma$. If the resulting test deck detects all $N! - 1$ original swaps, the algorithm finishes. If it fails to detect a specific swap, an "oracle" problem (CUT) identifies the undetected swap, adds it to the subset, and the problem is re-solved.

We made this process highly efficient in practice by reformulating the steps into mixed-integer linear optimization problems and introducing five key algorithmic improvements:

- **Reducing Decision Variables:** Dynamically identifying and removing unnecessary constraints and variables to lower per-iteration computation cost.

- **Enforcing Structure (Within Contests):** Adding constraints that force candidates in the same contest to receive a strictly increasing number of votes, vastly reducing the solution space.
- **Enforcing Structure (Across Contests):** Lexicographically ordering distinct vote counts across similar, equivalent contests to avoid redundant calculations.
- **Finding High-Quality Cuts:** Guiding the algorithm to find “minimal" non-identity bijections that eliminate the largest possible number of invalid test decks in each iterative step.
- **Combining Noncompetitive Contests:** Grouping uncontested races into a single contest to drastically decrease the number of algorithmic iterations needed.

Real-World Deployment: Michigan’s 2022 General Election

To prove the viability of this approach for real-world election administration, we partnered with the Michigan Bureau of Elections and retrospectively applied our algorithm to all 6,928 ballot styles used in the state’s November 2022 general election. The success of the deployment hinged on two practical factors: the length of the newly generated test decks and the computation time required to generate them.

1. Minimal Increase in Ballot Length

Long test decks pose high financial and operational costs for election officials. A priori, one might anticipate that decks providing mathematical security guarantees would be significantly longer than heuristic-based ones.

Remarkably, our results showed that the optimally secure test decks required, on average, only 1.2% more ballots than the heuristic-based test decks Michigan would otherwise use. In fact, for all but 493 of the 6,928 ballot styles, our RLAT test decks required the exact same number of ballots as current heuristic practices (see Figure 3). The slight increase in the remaining styles was dictated purely by the mathematical necessity of distinguishing candidates in noncompetitive contests.

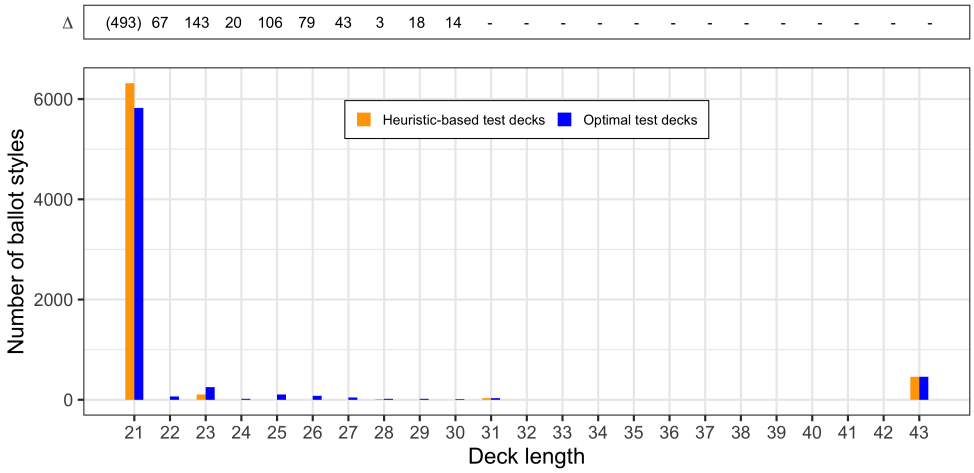


Figure 3: Distributions of lengths of heuristic-based test decks versus optimal RLAT test decks. The near-identical distribution proves the approach requires only minor increases in deck length.

2. Highly Practical Computation Times

Election preparation operates on a strict schedule, requiring optimal test decks for an entire state to be generated within 24 to 48 hours. By applying our cutting-plane method and utilizing strategies to reuse optimal solutions across similar ballot styles, we computed the optimal test decks for all 6,928 Michigan ballot styles in less than seven hours on a standard home computer. For 90% of the algorithmic invocations, computation took less than 2.5 minutes.

Conclusion

By applying robust optimization to Logic and Accuracy Testing, we transform a century-old intuition-based procedure into a modern, rigorous pre-election defense against misconfiguration cyberattacks. Our results demonstrate that election administrators can achieve mathematically proven security guarantees without facing significant administrative burdens or meaningfully increasing the size of their test decks.

Following this research, the RLAT approach was piloted in real-world elections by the Michigan Bureau of Elections during the summer of 2023. It offers a low-cost, highly effective way to safeguard election integrity, ensuring that any vote-swapping anomaly is caught well before a single real ballot is cast, ultimately bolstering public confidence in democratic institutions.

References

[1] Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski. *Robust Optimization*, volume 28. Princeton University Press, 2009.

[2] Dimitris Bertsimas, David B Brown, and Constantine Caramanis. Theory and applications of robust optimization. *SIAM Review*, 53(3):464–501, 2011.

[3] Nick Corasaniti. A Pennsylvania county's election day nightmare underscores voting machine concerns. *New York Times*, November 2019.

[4] PLLC DePerno Law Office. Dominion. URL https://www.depernolaw.com/dominion.html.

[5] Georgia Voter Registration & Elections - DeKalb County. Board of registration and elections meeting notes, July 2022. URL https://www.dekalbcountyga.gov/sites/default/files/users/user3597/Board%20Materials%202022-07-14.pdf.

[6] J. Alex Halderman. Analysis of the Antrim county, Michigan November 2020 election incident. In *Proceedings of the 31st USENIX Security Symposium*, pages 589–605. USENIX Association, 2022.

[7] Josiah Walker, Nakul Bajaj, Braden L Crimmins, and J Alex Halderman. Logic and accuracy testing: A fifty-state review. In *Electronic Voting: Seventh International Joint Conference*, volume 13553 of *Lecture Notes in Computer Science*. International Joint Conference on Electronic Voting, Springer, September 2022.

Research Highlight
Ambiguous Dynamic Treatment Regimes: A Reinforcement Learning Approach

by SOROUSH SAGHAFIAN
HARVARD UNIVERSITY, CAMBRIDGE MA 02138
SOROUSH_SAGHAFIAN@HKS.HARVARD.EDU

1 Introduction

A central objective in precision medicine, public policy, and economics is to discover *when* and *how* to dynamically make decisions affecting individuals—personalized to their evolving characteristics—so as to maximize outcomes over time. Dynamic Treatment Regimes (DTRs) formalize this objective: a DTR is a sequence of decision rules that maps a subject's observable history to a recommended action at each decision epoch [12, 22]. When learning and optimizing them using observational data, however, DTRs require *sequential ignorability*—the assumption that, conditional on measured covariates, treatment assignment is independent of all future potential outcomes [13, 14]. In various real-world applications, this assumption is almost never verifiable and is routinely violated. In healthcare, for example, unmeasured disease severity, patient adherence, or physician intuition can influence both the treatment selected and the subsequent outcome.

The challenge is compounded when unmeasured confounders are *time-varying*, meaning that they could be affected by previous decisions. In such settings, naïve adjustment methods that do not account for this feedback create additional bias [15]. Even if one postulates a specific causal model for the dynamics of unobserved confounders and their impact on observed data— the data generating model—such a model is itself subject to substantial ambiguity, and hence, any useful data-driven learning algorithm should be able to handle *model ambiguity* [see, also [19, 21]].

Saghafian [17] addresses both challenges simultaneously. The paper makes four inter-related contributions that together form a coherent and practically important framework for learning suitable dynamic decisions from data:

1. It extends DTRs to *Ambiguous* DTRs (ADTRs) in which the causal impact of any treatment policy is evaluated over a *cloud* of plausible data-generating models rather than a single assumed model.
2. It connects ADTRs to Ambiguous Partially Observable Markov Decision Processes (APOMDPs) [16], reinterpreting time-varying unobserved confounders as latent states with ambiguous transition dynamics.
3. It develops two efficient off-policy Reinforcement Learning (RL) algorithms—*Direct Augmented V-Learning* (DAV-Learning) and *Safe Augmented V-Learning* (SAV-Learning)—that learn optimal treatment regimes from observational data under model ambiguity.
4. It introduces the concept of *two-way personalization*: the resulting treatment policies are tailored simultaneously to the individuals' (e.g., patients) varying context variables and to the decision-maker's (e.g., a physician's) behavioral attitude toward ambiguity, parameterized by a pessimism level $\alpha \in [0, 1]$.

The advantages of the proposed framework in Saghafian [17] is investigated via a case study of New Onset Diabetes After Transplantation (NODAT), a condition that arises from the diabetogenic side-effect of immunosuppressive drugs (e.g., tacrolimus) given to organ transplant recipients [1, 2, 8, 9, 10]. Physicians must jointly manage the risk of organ rejection (requiring high tacrolimus) and the risk of diabetes onset (requiring low tacrolimus and possibly insulin). Clinical data on 407 kidney transplant patients (over 63,000 observations) collected at the Mayo Clinic provide the empirical testbed.

2 The ADTR Framework

Let $(O_t, A_t)_{t\in\mathcal{T}}$ denote the observed covariates and actions at each decision epoch. Let S_t denote the unobserved (latent) health state at time t. A treatment regime $\lambda = (\lambda_t)_{t\in\mathcal{T}}$ maps the observable history $H_t^o = (O_1, A_1, \ldots, O_t)$ to a probability distribution over actions. The overall gain under λ is the discounted

sum of immediate gains $G_t = g(S_t, A_t)$:

$$\Gamma_T(\lambda) = \sum_{t \in \mathcal{T}} \beta^{t-1} G_t^{\lambda}, \quad \beta \in (0,1).$$

In a standard DTR, one estimates the optimal λ^* by maximizing $\mathbb{E}[\Gamma_T(\lambda)]$ over the observed data, implicitly committing to a single causal model, and hence, a unique imposed distribution for $\Gamma_T(\lambda)$. In ADTRs [17], the causal model is not uniquely identified from the observed data. The set $\mathcal{M}$ of all models consistent with the observations constitutes the *ambiguity set*. Under each model $m \in \mathcal{M}$, the gain $\Gamma_T(\lambda)$ considered as a potential outcome variable of interest $Y(\lambda) = \Gamma_T(\lambda)$ follows a different distribution f^m. To compare policies, thus, Saghafian [17] adopts the α-Maximin Expected Utility (α-MEU) criterion [5, 6, 16]:

$$\mathrm{MEU}_\alpha[Y(\lambda)] = \alpha \inf_{f^m \in \hat{\mathcal{F}}} \mathbb{E}_{f^m}[Y(\lambda)] + (1-\alpha) \sup_{f^m \in \hat{\mathcal{F}}} \mathbb{E}_{f^m}[Y(\lambda)], \quad \alpha \in [0,1].$$

Here $\alpha = 1$ recovers the classical worst-case (maximin) criterion, $\alpha = 0$ is the optimistic (maximax) criterion, and intermediate values interpolate between the two. When $|\mathcal{M}| = 1$, MEU_α reduces to the standard expectation, so the traditional DTR optimality criterion is a special case.

This formulation is important for two reasons. First, it avoids the well-documented over-conservatism of the pure maximin view, which Savage (1951) described as "ultrapessimistic" [17]. Second, and crucially for real-world applications, it allows the decision-maker's own attitude towards ambiguity to be encoded in α, yielding one the paper's central conceptual contribution: *two-way personalization*. A decision-maker (e.g., a physician) who is particularly averse to worst-case outcomes (high α) will receive different recommendations than one who is more optimistic (low α). When needed, however, α can be viewed as a tuning parameter and optimized to gain the best overall recommendation.

Connection to APOMDPs. When the dynamics of the state variables satisfy the Markov property, the ADTR problem maps cleanly onto an Ambiguous Partially Observable Markov Decision Process (APOMDP) [16]. In this mapping, the unobserved confounders S_t become the latent state of the POMDP, while the observable history H_t^o is used to form a Bayesian belief distribution π_t over S_t via a standard belief-updating operator [16, 17]. The ambiguity set $\mathcal{M}$ translates into multiple candidate transition and emission probability matrices (P_m^a, Q_m^a). Crucially, Saghafian [16] has shown that the APOMDP value function $V_t(\pi)$ under some conditions is *piecewise linear and continuous* in the belief π—a structural property that the paper exploits to design computationally tractable data-driven learning algorithms.

Generalized Sequential Importance Sampling. Before turning to the Markovian case, Saghafian [17] establishes a non-Markovian baseline approach termed *Generalized Sequential Importance Sampling* (GSIS). GSIS reweights observed trajectories by importance sampling weights $w_t(\lambda^e) = \frac{\lambda_t^e(A_t|H_t^o)}{\lambda_t^b(A_t|H_t^o)}$, where λ^b is the behavior (data-generating) policy and λ^e is any policy to be evaluated. Under sequential ignorability and almost-sure overlap between evaluation and behavior policies, GSIS yields an MEU_α-unbiased estimator of $\Gamma_T(\lambda^e)$. The paper also extends GSIS to the practically important case of *Bounded Unobservable Confounding* (BUC), where the sequential ignorability no longer holds, but unobserved confounders affect treatment propensities only within known multiplicative bounds $\eta_t^m \in [1, \infty)$, providing approximate unbiased estimation even when sequential ignorability fails.

Finally, under Markovian behavior, [17] demonstrates that causal relations in observed ADTR data can be represented via a directed acyclic graph (DAG) depicted in Fig. 1, which is a DAG representation of APOMDPs. This enables developing causal RL algorithms as discussed in the next section.

3 Theoretical Results

Weight-Adjusted Bellman Equations. A central technical result in Saghafian 17 is based on a *weight-adjusted Bellman equation* for the value function under an evaluation policy μ^e in any single POMDP model m:

$$V_{T-t+1}^{m,\mu^e}(\pi_t) = \mathbb{E}_m \left[\frac{\mu_t^e(A_t|\Pi_t^m)}{\mu_t^b(A_t|\Pi_t^m)} \Big(G_t + \beta V_{T-t}^{m,\mu^e}(T(\Pi_t^m, A_t, O_t, m)) \Big) \,\Big|\, \Pi_t^m = \pi_t \right].$$

This equation—which holds whenever the behavior policy satisfies positivity and sequential ignorability after conditioning on belief states—is fundamental: it expresses the value of any evaluation policy purely in terms of observable quantities, enabling estimation from data.

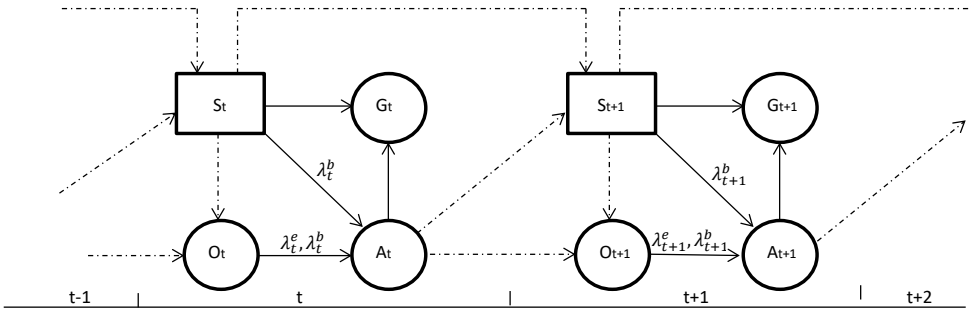


Figure 1: DAG representation of APOMDPs (Fig. 1 in [17]). *Circles:* observable variables; *Rectangles:* unobservable variables; *Solid arrows:* unambiguous causal mechanisms; *Dashed arrows:* ambiguous causal mechanisms.

Under BUC, the equation generalizes to sandwich bounds on the value function using weight modifiers described in [17] that incorporate the confounding bounds η_t^m. This BUC extension underpins the DAV-Learning-BUC and SAV-Learning-BUC variants of the main algorithms.

Reinforcement Learning Algorithms: DAV-Learning and SAV-Learning. Both algorithms learn the value function in the class $\mathcal{V}$ of piecewise linear and continuous functions on the belief simplex Δ_S, using a parametric form $V_\infty^{m,\mu^e}(\pi; c) = (b(\pi))'c$ where $b(\pi)$ is a predefined basis (c is used here in place of ψ in [17] for the ease of notation). The algorithms differ in *when* they incorporate model ambiguity.

DAV-Learning (Algorithm 1 in [17]) first estimates the value function separately for each model $m \in \mathcal{M}$ via regularized minimization of the empirical Bellman residual:

$$\hat{c}_n^{m,\mu^e} = \arg\min_{c \in \mathcal{C}} \big\{ (\varphi_n^{m,\mu^e}(c))' \Omega \, \varphi_n^{m,\mu^e}(c) + \theta_n P(c) \big\},$$

and then combines model-specific gains via the α-MEU operator at the end of the horizon:

$$\hat{\Gamma}_\infty(\mu^e) = \alpha \inf_{m \in \mathcal{M}} \hat{\Gamma}_\infty^m(\mu^e) + (1-\alpha) \sup_{m \in \mathcal{M}} \hat{\Gamma}_\infty^m(\mu^e).$$

SAV-Learning (Algorithm 2 in [17]) instead incorporates ambiguity *up front* by estimating the value function parameter as the α-MEU of the individual model estimates:

$$\hat{c}_n^{\mu^e} = \alpha \, \hat{c}_n^{\underline{m},\mu^e} + (1-\alpha) \, \hat{c}_n^{\overline{m},\mu^e},$$

where $\underline{m}$ and $\overline{m}$ minimize and maximize the parameter norm over $\mathcal{M}$. This "safe" integration of ambiguity at the estimation stage makes the resulting policy more robust to the choice of α, at the cost of some mean performance relative to DAV-Learning.

Asymptotic Properties of the Algorithms. The paper establishes the asymptotic behavior of both algorithms under five regularity conditions covering the parameter space, the policy space, the trajectory process, and the model space. The key results (Theorems 1 and 2 in [17]) are:

1. **Weak consistency**: For any fixed evaluation policy μ^e and model m, $\hat{c}_n^{m,\mu^e} \xrightarrow{p} c_*^{m,\mu^e}$ and $\hat{\Gamma}_\infty(\mu^e) \xrightarrow{p} \Gamma_\infty(\mu^e)$.
2. **Asymptotic normality**: $\sqrt{n}(\hat{c}_n^{m,\mu^e} - c_*^{m,\mu^e})$ converges in distribution to a zero-mean Gaussian process in $\ell^\infty(\mathcal{M})$.
3. **Consistency of the optimal policy**: $d_{\mathcal{M}}(\hat{\mu}_n^{e*}, \mu^{e*}) \xrightarrow{p} 0$ and $\hat{\Gamma}_\infty(\hat{\mu}_n^{e*}) \xrightarrow{p} \Gamma_\infty(\mu^{e*})$.

A notable technical feature is that the proofs must handle both non-i.i.d. trajectory data (which are absolutely regular stationary processes with β-mixing coefficients satisfying a summability condition) and the presence of multiple ambiguous models. The asymptotic results in [17] leverage findings from *empirical process theory* for dependent data [3, 7] to establish *Donsker properties* in $\ell^\infty(\mathcal{M})$.

The tuning parameter θ_n must satisfy $\theta_n = o_p(n^{-1/2})$, which is the standard rate for penalized estimators that achieve consistency and asymptotic normality simultaneously. These results guarantee that with enough data, ADTRs can be reliably estimated and that uncertainty in the estimated optimal policy diminishes at the usual $\sqrt{n}$ rate.

4 Empirical Results

Case Study: NODAT at Mayo Clinic. The clinical data set comprises 407 kidney transplant recipients observed monthly for 12 months post-transplant (13 covariates, 9 latent health states, 4 possible actions combining tacrolimus dosing and insulin use). After cubic spline interpolation to fill data gaps, the full data set contains 63,492 observations. The ambiguity set $\mathcal{M}$ contains 4 models constructed via an entropy ball around Baum–Welch estimates of the POMDP emission and transition matrices.

Table 5 of Saghafian [17] reports total discounted gains (discount factor $\beta = 0.95$) under the observed clinical regime and the two proposed algorithms across five values of $\alpha \in \{0.00, 0.25, 0.50, 0.75, 1.00\}$. The main findings are:

- **Both algorithms substantially outperform observed clinical practice.** DAV-Learning improves over the observed regime by 10%–42% and SAV-Learning by 10%–32%, depending on α.
- **DAV-Learning achieves higher mean gains**, particularly for low α (optimistic) settings. At $\alpha = 0$, DAV-Learning yields a mean total gain of 2.085 versus 1.472 under the observed regime.
- **SAV-Learning is far more robust to the choice of α.** Whereas DAV-Learning performance degrades markedly as α increases from 0 to 1 (from 2.085 to 1.609), SAV-Learning spans a narrower range (1.949 to 1.606). An optimizer using SAV-Learning need not carefully tune α.

These findings reflect the structural difference between the algorithms: SAV-Learning's "safe" up-front aggregation of model ambiguity effectively buffers against the choice of pessimism level, while DAV-Learning's end-of-horizon aggregation preserves sensitivity to α but also unlocks higher upside for optimistic decision-makers.

Synthetic Data Experiments. Synthetic experiments simulate 100 patients over 10 periods under $|\mathcal{M}| = 10$ misspecified models generated from Dirichlet distributions with random parameters, with a known true model for benchmarking. Key results (Table 6 in [17]) mirror the clinical findings: both DAV-Learning and SAV-Learning improve over the observed regime, and SAV-Learning's performance is more stable across α values.

Robustness to Model Ambiguity. The most striking empirical result concerns robustness. The paper benchmarks all algorithms against an imaginary oracle who knows both the true data-generating model and the optimal policy under that model. The *gain loss* (regret relative to the oracle) exhibits a U-shaped curve in α: both extreme pessimism ($\alpha = 1$) and extreme optimism ($\alpha = 0$) are suboptimal choices, while intermediate values (around $\alpha \approx 0.25$) minimize regret. More importantly, the maximum gain loss across all values of α for all four algorithms remains below 0.6%. This is a remarkable finding: a decision-maker operating under complete uncertainty about the data-generating process can, by using DAV-Learning or SAV-Learning, achieve performance essentially indistinguishable from an oracle who knows the truth.

This result in [17] provides concrete empirical support for the paper's philosophical position: a "cloud of models" approach to causal inference—taking a middle ground between fully model-based and fully model-free reasoning—can simultaneously deliver personalization, interpretability, and robustness.

5 Conclusions

The core insight in Saghafian [17]—that model ambiguity is not merely a nuisance to be minimized but an inherent feature of observational data that should be *built into* the learning framework—makes a significant and timely contribution to the intersection of causal inference, reinforcement learning, and sequential decision-making. It can reorients how researchers might think about off-policy evaluation under unobserved confounding. The paper's "cloud of models" methodology explicitly positions ADTRs between fully structural causal models (which commit to a single data-generating process) and purely nonparametric, model-free methods (which make no structural assumptions). This middle ground echoes calls by Manski [11] and others for causal inference methods that remain valid across the set of all feasible models.

References

[1] Boloori, A., Saghafian, S., Chakkera, H. A., and Cook, C. B. (2015). Characterization of remitting and relapsing hyperglycemia in post-renal-transplant recipients. *PLoS One,* 10(11), e0142363.

[2] Boloori, A., Saghafian, S., Chakkera, H. A., and Cook, C. B. (2020). Data-driven management of post-transplant medications: An ambiguous partially observable Markov decision process approach. *Manufacturing and Service Operations Management*, 22(5), 1066-1087.

[3] Dedecker, J., and Louhichi, S. (2002). Maximal inequalities and empirical central limit theorems. In *Empirical Process Techniques for Dependent Data*, pp. 137–159. Birkhäuser, Boston.

[4] Frank, R. G., and Zeckhauser, R. J. (2007). Custom-made vs. ready-to-wear treatments: Behavioral propensities in physicians' choices. *Journal of Health Economics*, 26(6):1101–1127.

[5] Ghiradato, P., Maccheroni, F., and Marinacci, M. (2004). Differentiating ambiguity and ambiguity attitude. *Journal of Economic Theory*, 118:133–173.

[6] Hurwicz, L. (1951). Optimality criteria for decision making under ignorance. Cowles Commission Discussion Paper: Statistics No. 370.

[7] Kosorok, M. R. (2008). *Introduction to Empirical Processes and Semiparametric Inference*. Springer, New York.

[8] Munshi, V. N., Saghafian, S., Cook, C. B., Steidley, D. E., Hardaway, B., and Chakkera, H. A. (2020). Incidence, risk factors, and trends for postheart transplantation diabetes mellitus. *The American Journal of Cardiology*, 125(3), 436-440.

[9] Munshi, V. N., Saghafian, S., Cook, C. B., Werner, K. T., and Chakkera, H. A. (2020). Comparison of post-transplantation diabetes mellitus incidence and risk factors between kidney and liver transplantation patients. *PloS one*, 15(1), e0226873.

[10] Munshi, V. N., Saghafian, S., Cook, C. B., Aradhyula, S. V., and Chakkera, H. A. (2021). Use of imputation and decision modeling to improve diagnosis and management of patients at risk for new-onset diabetes after transplantation. *Annals of Transplantation*, 26, e928624-1.

[11] Manski, C. F. (2021). Econometrics for decision making: Building foundations sketched by Haavelmo and Wald. *Econometrica*, 89(6):2827–2853.

[12] Murphy, S. A. (2003). Optimal dynamic treatment regimes. *Journal of the Royal Statistical Society: Series B*, 65(2):331–355.

[13] Murphy, S. A., van der Laan, M. J., Robins, J. M., and CPPRG (2001). Marginal mean models for dynamic regimes. *Journal of the American Statistical Association*, 96(456):1410–1423.

[14] Robins, J. (1986). A new approach to causal inference in mortality studies with a sustained exposure period. *Mathematical Modelling*, 7(9–12):1393–1512.

[15] Robins, J., Hernán, M. A., and Brumback, B. (2000). Marginal structural models and causal inference in epidemiology. *Epidemiology*, 11(5):550–560.

[16] Saghafian, S. (2018). Ambiguous partially observable Markov decision processes: Structural results and applications. *Journal of Economic Theory*, 178:1–35.

[17] Saghafian, S. (2024). Ambiguous dynamic treatment regimes: A reinforcement learning approach. *Management Science*, 70(9):5667–5690.

[18] Saghafian, S., and Murphy, S. A. (2021). Innovative healthcare delivery: The scientific and regulatory challenges in designing mHealth interventions. *NAM Perspectives*, National Academy of Medicine, Washington, DC.

[19] Saghafian, S., and Tomlin, B. (2016). The newsvendor under demand ambiguity: Combining data with moment and tail information. *Operations Research*, 64(1), 167-185.

[20] Saghafian, S., Tomlin, B., and Biller, S. (2022). The Internet of Things and information fusion: Who talks to who? *Manufacturing & Service Operations Management*, 24(1):333–351.

[21] Saghafian, S. (2025). *Insight-driven Problem Solving: Analytics Science to Improve the World*. Cambridge University Press.

[22] Tsiatis, A. A., Davidian, M., Holloway, S. T., Laber, E. B., and Kosorok, M. R. (2019). *Dynamic Treatment Regimes: Statistical Methods for Precision Medicine*. Chapman and Hall/CRC, Boca Raton, FL.

Research Highlight
Bandits atop Reinforcement Learning: Tackling Online Inventory Models with Cyclic Demands

by Evelyn Xiao-Yue Gong[1], David Simchi-Levi[2],
[1]Carnegie Mellon University
[2]Massachusetts Institute of Technology

1 Introduction

Sequential decision making under uncertainty is a recurring theme in modern computing and operations. Inventory management is a natural and practically important sequential decision problem. In a periodic-review model, a retailer repeatedly observes inventory, chooses an order quantity, receives demand, and pays holding, lost-sales, or backlogging costs. Classical inventory theory gives sharp structural results when the demand distribution is known; in practice, however, the distribution must often be learned while the system is already operating. A central difficulty is that real demand is usually not identically distributed over time. Many products exhibit weekly, monthly, seasonal, or event-driven cycles. A product may sell differently on Mondays and Saturdays; a drugstore may experience systematic weekend effects; a restaurant may face recurring delivery and consumption patterns. Treating such demand as i.i.d. is computationally convenient, but it can lead to systematically wrong inventory levels.

Our work studies online inventory control with unknown cyclic stochastic demand distributions. The retailer may know the cycle length H, but not the demand distributions within the cycle. The goal is to design learning policies that obtain minimal regret compared to a clairvoyant policy that knows all demand distributions in advance. The main methodological message is that sharper guarantees are possible when the algorithm exploits the structure of the operations model rather than treating it as a black-box Markov decision process.

The key structure is feedback. In backlogging models, the feedback is full: one realized demand sample reveals the counterfactual outcome for every feasible replenishment decision. In lost-sales models, the feedback is one-sided: ordering up to y reveals what would have happened for all lower levels $y' \leq y$. Our work utilizes these observations to design reinforcement learning algorithms whose regret bounds avoid the usual dependence on the cardinality of the state-action space, which is expected in the case of full feedback, but somewhat surprising in the case of one-sided feedback. Thus the algorithms are not merely applications of Q-learning to inventory; they are examples of how domain-specific counterfactual feedback can be converted into statistical and computational efficiency.

2 Cyclic inventory learning and regret

Consider a horizon of $T = KH$ periods partitioned into K cycles of length H. Demand in period t is generated from a distribution F_h, where $h \in \{1, \ldots, H\}$ denotes the position of t within its cycle. Thus demands are independent but not identically distributed: the distribution at a fixed position h repeats across cycles, while the H distributions inside a cycle may be arbitrary. The cycle length H is treated as a known constant. This is natural when the cycle corresponds to a weekly, monthly, or seasonal pattern.

In the lost-sales model, the retailer begins period t with inventory x_t, chooses an order-up-to level $y_t \geq x_t$, and then observes sales $\min\{y_t, D_t\}$. Unsatisfied demand is lost. If $o_h > 0$ is the holding cost and $p_h > 0$ is the lost-sales penalty in period type h, the one-period cost is

$$C_h(y_t, D_t) = o_h(y_t - D_t)^+ + p_h(D_t - y_t)^+.$$

The lost part $(D_t - y_t)^+$ is not observed, so the realized cost is not directly available to the learner. A standard reduction replaces it by the observable pseudo-cost

$$\widetilde{C}_h(y_t, D_t) = o_h(y_t - D_t)^+ - p_h \min\{y_t, D_t\}.$$

The difference between C_h and $\widetilde{C}_h$ is $p_h D_t$, which is independent of the decision y_t. Hence optimizing regret with respect to pseudo-cost is equivalent to optimizing regret with respect to the true cost.

For a learning policy π, regret is the expected performance gap against the clairvoyant optimal policy:

$$\mathrm{Regret}_\pi(T) = \mathbb{E}\left[\sum_{t=1}^{T} C_t^\pi\right] - \mathbb{E}\left[\sum_{t=1}^{T} C_t^{\mathrm{OPT}}\right].$$

In the lost-sales model with zero lead time, cyclic base-stock policies are proved to be optimal: there are H target inventory levels $B_1, \ldots, B_H$, and in period type h the policy orders up to B_h whenever feasible. Thus learning the optimal policy amounts to learning the correct base-stock level at each position in the cycle. The difficulty is that these levels interact dynamically through leftover inventory.

The paper distinguishes between two versions of each inventory model. In the episodic version, leftover inventory is discarded or salvaged at the end of each cycle. This resets the system and makes the model an episodic Markov decision process. In the nondiscarding version, inventory carries over across cycles, as in standard nonperishable inventory models. The nondiscarding version is operationally more important, but the episodic version is a crucial building block for learning.

3 Why cyclic demand is algorithmically different

The cyclic setting is not a cosmetic modification of the i.i.d. demand setting. Several approaches that work for i.i.d. inventory learning no longer apply. First, the expected cost of holding a fixed inventory level depends on the position in the cycle. Second, the relevant optimization landscape can lose the convexity or one-dimensional structure exploited by stochastic-gradient or bandit methods for i.i.d. demand. Third, the action selected in one period affects the feasible action set in later periods through leftover inventory. These features make the problem genuinely sequential.

A generic reinforcement learning formulation is possible. The inventory level is the state, the order-up-to level is the action, and the period within the cycle is the time index. However, generic tabular reinforcement learning would produce regret bounds depending on the number of states and actions, which is unsatisfactory because inventory levels are often naturally large or continuous and are discretized only for analysis.

The paper's key insight is that inventory models provide richer feedback than a generic Markov decision process. In the lost-sales model, feedback is one-sided. If the retailer orders up to y, then after observing sales $\min\{y, D\}$, she can infer what the pseudo-cost and next inventory would have been for every smaller order-up-to level $y' \leq y$. Thus choosing a high base-stock level provides information about all lower base-stock levels. But choosing too high a level may cause excessive holding cost and may also leave too much inventory for future periods. This creates a nontrivial exploration–exploitation tradeoff.

In the backlogging model, the feedback is even richer. Demand is observed, and unmet demand is backlogged rather than lost. Once demand is realized, the learner can compute what the cost and next state would have been for every feasible replenishment decision. This is a full-feedback structure. It allows a more direct adaptation of Q-learning and leads to a more straightforward regret analysis.

The majority of the paper assumes that the cycle length is known a priori. The paper also discusses how the results extend to the setting where the cycle length is unknown.

4 Algorithms

4.1 HQL for episodic lost sales

The first algorithm, Elimination-Based Half-Q-Learning (HQL), is designed for the episodic single-product lost-sales model. It combines Q-learning with elimination over candidate base-stock levels. For each period type h, the algorithm maintains a running set A_h^k of base-stock levels that have not yet been statistically ruled out by episode k. To exploit the lower-sided feedback structure, HQL chooses the largest feasible action in the current running set. This maximizes the amount of counterfactual information observed in that period, because all lower base-stock levels can be evaluated from the same realized sales observation.

Conditional on choosing a feasible base-stock level, the reward and next inventory depend on the chosen level and the realized demand, but not on the previous inventory. Thus the relevant Q-value can be written as $Q_h(y)$ rather than $Q_h(x, y)$, although feasibility still depends on x.

A technical subtlety is that high inventory can prevent the learner from choosing from the running set in a later period. HQL handles this by using stopping times. At the end of each episode, the algorithm reconstructs counterfactual trajectories under alternative base-stock levels, but updates Q-values only up to the next time when the running set again becomes feasible. This delayed Bellman update is what allows one-sided feedback to be used without losing control of the sequential dependence.

The resulting regret bound for the episodic lost-sales model is

$$\tilde{O}\Big(H^3\sqrt{T}\Big).$$

Note that this regret bound is independent of the cardinality of the state-action space. The paper also proves a matching $\Omega(H\sqrt{T})$ lower bound, up to problem-parameter and logarithmic factors. This shows that the $\sqrt{T}$ dependence is information-theoretically optimal.

4.2 Meta-HQL: bandits atop reinforcement learning

The main nondiscarding lost-sales result is obtained by placing a bandit algorithm on top of shifted copies of HQL. The structural observation, adapted from the cyclic inventory literature, is that if a cycle is shifted so that the largest optimal base-stock level occurs at the first period of the episode, then the nondiscarding lost-sales problem is equivalent, for learning purposes, to an episodic problem with that shifted cycle. Intuitively, if the first base-stock level is the largest one, then carrying inventory across the end of the cycle does not change the cost relative to discarding, since the next period is precisely the one at which the system is willing to raise inventory to the largest target level.

The learner does not know which period type has the largest optimal base-stock level. Meta-HQL treats each possible shift $w \in \{1, \dots, H\}$ as an arm in a bandit problem. Pulling arm w means running the w-shifted version of HQL for a block of episodes. The algorithm proceeds in phases. In each phase it runs each surviving arm for a geometrically increasing number of episodes, estimates its average reward, and eliminates arms that are statistically inferior. Switching between arms is handled by temporarily ordering no replenishment until inventory drops enough for the next shifted HQL copy to operate.

This two-level construction yields

$$\tilde{O}\Big(H^{3.5}\sqrt{T}\Big)$$

regret for the nondiscarding single-product lost-sales model with zero lead time. The important qualitative point is again the absence of the state-action-space cardinality in the regret bound. The paper proves a matching $\Omega(H\sqrt{T})$ lower bound for this nondiscarding model as well, by constructing an instance whose optimal base-stock sequence makes the nondiscarding and episodic versions equivalent.

4.3 FQL for episodic multiproduct backlogging

The third algorithm, Full-Q-Learning (FQL), applies to episodic multiproduct backlogging models. Here the state includes inventory and pipeline replenishment for multiple products, and the action may be a vector of replenishment quantities subject to order limits and fixed joint-ordering costs. Cyclic base-stock policies are not necessarily optimal in this general model, so the algorithm learns over the full state-action space.

The full-feedback property is enpowering. After demand is realized, the learner can compute the reward and next state associated with every state-action pair at that time step. Consequently, FQL can update all relevant Q-values in each episode. This removes the need for optimism bonuses and avoids the dependence on the cardinality of the state-action space that appears in generic Q-learning regret bounds. For the n-product episodic backlogging model with lead time L, the regret is of order

$$\tilde{O}\Big(H^2\sqrt{T}\,\Big).$$

The absence of a state-action-space factor is less surprising than in the one-sided-feedback setting.

4.4 Mimic-QL for nondiscarding multiproduct backlogging

The general nondiscarding multiproduct backlogging model lacks the base-stock structure used by Meta-HQL. The paper therefore introduces Mimic-QL. The idea is to simulate an episodic problem with a much

Model	Feedback	Algorithm	Regret order
Episodic lost sales	One-sided	HQL	$\tilde{O}(\sqrt{T})$
Episodic multiproduct backlogging	Full	FQL	$\tilde{O}(\sqrt{T})$
Nondiscarding lost sales	One-sided	Meta-HQL	$\tilde{O}(\sqrt{T})$
Nondiscarding multiproduct backlogging	Full	Mimic-QL	$\tilde{O}(T^{5/6})$

Table 1: Summary of the main algorithmic guarantees. The $\tilde{O}(\sqrt{T})$ bounds for episodic lost sales, episodic backlogging, and nondiscarding lost sales match lower bounds up to these suppressed factors.

longer artificial cycle length J, where J is chosen as a multiple of the true cycle length H. Since a sequence that is cyclic with period H is also cyclic with period J, FQL can be run on this intermediate episodic model and is shown to have a $\tilde{O}(T^{5/6})$ regret guarantee.

5 Empirical evidence

The theoretical model is motivated by practical cyclic demand. The paper evaluates the algorithms on both synthetic cyclic data and a public Rossmann drugstore sales data set. Rossmann is a large European drugstore chain, and the data contain daily sales for more than one thousand stores over multiple years.

On the synthetic data, the experiments compare Meta-HQL with two benchmarks. The first is a clairvoyant optimal cyclic base-stock policy that has knowledge of all the realized demands a priori. The second is the best base-stock policy in hindsight that treats demand as i.i.d. across time. Meta-HQL rapidly converges toward the optimal cyclic policy, while the best i.i.d. base-stock level remains far from optimal. On the Rossmann data, with similar benchmarks, the same qualitative pattern appears. Meta-HQL initially incurs larger costs because it explores a broad range of base-stock levels, but its average cost then declines and approaches the near-optimal cyclic benchmark. The experiments also show how expert advice can be incorporated by shrinking the initial candidate sets. For example, historical upper bounds on demand can be used to reduce early exploration, improving short-term performance without changing the regret analysis.

The empirical message is that the cyclic structure is not merely a theoretical complication. Correctly modeling cyclicity can dominate even the best i.i.d. policy. The experiments also emphasize a useful operational point: regret-optimal algorithms can be initialized with domain knowledge and historical data, yielding much better finite-sample behavior while preserving the underlying online-learning framework.

6 Broader implications and concluding remarks

Several lessons extend beyond the inventory models analyzed in the paper. First, Markov decision processes arising in operations often have feedback structures that are absent from generic reinforcement learning formulations. One realized demand sample can reveal counterfactual information about many possible decisions. Algorithms that ignore this structure pay unnecessary statistical and computational costs.

Second, the distinction between one-sided and full feedback is algorithmically important. It is relatively straightforward to design Q-learning methods that utilize full feedback. One-sided feedback requires a more delicate balance: choosing larger actions improves information acquisition but can worsen immediate and future costs. HQL is designed precisely to manage this tradeoff.

Third, nondiscarding inventory illustrates a broader principle: a hard online learning problem can sometimes be decomposed into a collection of easier episodic problems plus a simultaneous higher-level model-selection problem. Meta-HQL is one instance of this idea. It does not simply apply a bandit algorithm to inventory decisions; rather, it applies bandit learning to choose among entire reinforcement learning algorithms, each corresponding to a different cyclic shift.

The paper also points to related applications with similar feedback. Airline overbooking has a one-sided structure: allowing a higher booking limit reveals information about what would have happened at lower limits. Online second-price auctions can have one-sided or full feedback depending on which bids are observed. Portfolio allocation has full feedback when the returns of all assets are observed after the allocation decision. These examples suggest that customized reinforcement learning for operations problems may ben-

efit from classifying feedback structures more explicitly.

References

[1] S. Agrawal and R. Jia. Learning in structured MDPs with convex cost functions: Improved regret bounds for inventory management. *Operations Research*, 70(3):1646–1664, 2022.

[2] X.-Y. Gong and D. Simchi-Levi. Bandits atop reinforcement learning: Tackling online inventory models with cyclic demands. *Management Science*, 70(9):6139–6157, 2024.

[3] W. T. Huh and P. Rusmevichientong. Online sequential optimization with biased gradients: Theory and applications to censored demand. *INFORMS Journal on Computing*, 26(1):150–159, 2014.

[4] C. Jin, Z. Allen-Zhu, S. Bubeck, and M. I. Jordan. Is Q-learning provably efficient? In *Advances in Neural Information Processing Systems*, 2018.

[5] H. Yuan, Q. Luo, and C. Shi. Marrying stochastic gradient descent with bandits: Learning algorithms for inventory systems with fixed costs. *Management Science*, 67(10):6089–6115, 2021.

[6] P. Zipkin. Critical number policies for inventory models with periodic data. *Management Science*, 35(1):71–80, 1989.

Research Highlight
Stepsize Hedging:
An Alternative Mechanism for Accelerating Gradient Descent

by JASON M. ALTSCHULER[1], PABLO A. PARRILO[2]
[1]PRINCETON
[2]LIDS - MASSACHUSETTS INSTITUTE OF TECHNOLOGY

1 Introduction

Consider the following question that arises in every introductory course on convex optimization:

What is the optimal choice of stepsizes for gradient descent?

Gradient descent (GD) is arguably the most foundational algorithm for continuous optimization and has a commensurately extensive literature. GD was first proposed by Cauchy in the 1800s and remains the workhorse in large-scale optimization applications across engineering, data science, and artificial intelligence. Yet, the fundamental question stated above—how to choose the sole parameter in GD—remained open even in seemingly simple convex settings, highlighting the potential for untapped algorithmic opportunities in those foundational settings and beyond.

Short or long stepsizes? Let us begin by recalling the definition of GD:

$$x_{t+1} = x_t - \alpha_t \nabla f(x_t)\,. \tag{1}$$

The basic idea is that the negative gradient is a descent direction, meaning that if one moves a small amount in this direction, then the value of the objective f decreases. This follows immediately by a Taylor expansion. The algorithmic question at hand is: *how far* should one move? A small stepsize α_t guarantees progress, but the progress will be correspondingly small. A large stepsize α_t is a natural idea to get more mileage out of the descent direction, but this may overshoot the optimum.

Mainstream prescription for smooth convex optimization: constant stepsize schedules. A natural way to quantify this tradeoff is to optimize the progress made in a single iteration. In the standard setup of smooth convex optimization, this one-step question has a well-known answer:

For one iteration of GD, there is a stepsize $\bar{\alpha}$ that achieves the fastest worst-case convergence rate. (2)

This stepsize is explicit.[1] This one-step guarantee underlies the textbook prescription: use the constant schedule $\alpha_t = \bar{\alpha}$ at every iteration t. See for example the textbooks [7, 10, 11, 24, 26, 30, 34].

However, even after optimizing $\bar{\alpha}$, this constant stepsize schedule may lead to slow convergence. This has motivated an extensive literature on *accelerated* first-order methods which famously modify GD by adding momentum or other internal dynamics, see the survey [18]. Many alternative stepsize schedules have also been proposed—for example exact line search, Armijo-Goldstein rules, Polyak-type schedules, and Barzilai-Borwein-type schedules—but none had led to an analysis that improves over the textbook constant-stepsize rate. The resulting conventional wisdom was that GD cannot be accelerated merely by changing its stepsizes.

Faster convergence via hedged stepsizes? Is this conventional wisdom correct? Surprisingly, the answer is no. The key observation is that optimality of the stepsize $\bar{\alpha}$ for one iteration does *not* imply optimality of repeating the stepsize $\bar{\alpha}$ for multiple iterations. The stepsize $\bar{\alpha}$ is chosen to protect against the worst case for a single iteration. But when GD is run for multiple iterations, the bad instances for different stepsizes need not be compatible with one another. A short step may be too conservative on one kind of instance (flat objectives f), while a long step may overshoot on another (steep objectives f); over several iterations, these opposing weaknesses can fail to align (since convex objectives f are rigid and cannot change curvature arbitrarily). This suggests a multi-step opportunity:

Can one combine stepsizes that are individually suboptimal
to obtain faster convergence over many iterations? (3)

We refer to this idea as *stepsize hedging*: rather than optimizing each step in isolation, choose a time-varying schedule that hedges between different worst-case behaviors.

[1]For example, $\bar{\alpha} = 1/M$ for M-smooth convex objectives, and $\bar{\alpha} = 2/(M+m)$ for objectives that are also m-strongly convex.

Motivation: the special case of quadratics. Time-varying stepsizes were initially explored (only) in the special case of convex quadratic optimization. A classic result due to Young in 1953 [49] shows that in this setting, the optimal stepsizes are non-constant and related to the roots of Chebyshev polynomials. This elegant result is recalled in detail in §2. However, many core phenomena in the quadratic setting do not extend beyond. For example, these Chebyshev stepsizes can make GD diverge beyond the special case of quadratics. Over the past 70 years, the continuous optimization community has devoted significant effort to developing alternative stepsize schedules beyond quadratic optimization. These are often empirically helpful. However, despite significant effort and many candidate stepsize schedules (even adaptive), there was no theoretical evidence that *any* stepsize schedule could lead to *any* speedup over the textbook GD convergence rate for (non-quadratic) convex optimization.

1.1 Main result

In the past decade, a line of work has shown that such an algorithmic opportunity persists: time-varying stepsizes can accelerate GD for (non-quadratic) convex optimization. The first such result was shown in the thesis [2] of the first author (advised by the second author), and an exciting flurry of ensuing work has sought to push this algorithmic opportunity to its limit. See the discussion of related work in §1.2.

We summarize here the results of the two papers [3, 4]. In these papers, we propose an unconventional stepsize schedule and show that it accelerates GD for smooth convex optimization. We term this the *silver stepsize schedule* due to the occurrence of the *silver ratio* $\rho = 1+\sqrt{2}$. It is explicit and non-adaptive, see §4 for the definition and a full discussion. Here we highlight the main results, namely the accelerated convergence rates that this enables for the convex and strongly-convex settings, respectively.

Below, for shorthand, we say that an iterative algorithm initialized at x_0 minimizes an M-smooth convex function f to ε error if its final iterate x_n satisfies $\frac{f(x_n)-f(x^*)}{M\|x_0-x^*\|^2} \leq \varepsilon$, where x^* denotes a minimizer of f. The factor of $M\|x_0 - x^*\|^2$ normalizes the performance to make it scale-invariant and meaningful.

Theorem 1 (Silver stepsizes for convex optimization [4])**.** *Fix any accuracy ε and smoothness parameter M. There exists a stepsize schedule of length*

$$n \asymp \varepsilon^{-\log_\rho 2} \approx \varepsilon^{-0.7864}$$

such that GD ε-minimizes any M-smooth convex function $f : \mathbb{R}^d \to \mathbb{R}$ in any dimension d.

Theorem 2 (Silver stepsizes for strongly convex optimization [3])**.** *Fix any accuracy ε, strong convexity parameter m, and smoothness parameter M. There exists a stepsize schedule of length*

$$n \asymp \kappa^{\log_\rho 2} \log \frac{1}{\varepsilon} \approx \kappa^{0.7864} \log \frac{1}{\varepsilon}$$

such that GD ε-minimizes any m-strongly convex, M-smooth function $f : \mathbb{R}^d \to \mathbb{R}$ in any dimension d. Here $\kappa = M/m \geq 1$ denotes the condition number.

For comparison, Theorem 1 improves the textbook rate of constant-step GD from $O(\varepsilon^{-1})$ to roughly $O(\varepsilon^{-0.7864})$. In the strongly convex setting, Theorem 2 analogously improves the textbook rate of GD from $O(\kappa \log \frac{1}{\varepsilon})$ to roughly $O(\kappa^{0.7864} \log \frac{1}{\varepsilon})$. These results establish stepsize hedging as an alternative mechanism for accelerating GD. Interestingly this does not match the fully accelerated rate obtained by modifying GD beyond stepsizes [31]. Our asymptotic rates are conjectured optimal among all (non-adaptive[2]) stepsize schedules.

1.2 Related work

There is an extensive literature on accelerated first-order algorithms, e.g., [18, 29, 30]. As opposed to the mainstream approach which modifies GD, we focus here on an alternative mechanism for accelerating GD: time-varying stepsize schedules.

As mentioned above, the benefit of time-varying stepsizes was classically known for the special case of quadratic optimization since Young in 1953 [49], but remained open beyond quadratics. The first such results were by Altschuler [2], who showed that time-varying hedging can lead to improved rates by analyzing

[2] A non-adaptive schedule is a predetermined sequence $\alpha_0, \alpha_1, \ldots$ that may depend on problem parameters such as M, m, n, but not on the observed iterates or gradients. In contrast, an adaptive schedule may choose α_t using information from earlier iterations.

	Quadratic	Convex
Mainstream stepsizes	$\Theta(\kappa)$ by constant stepsizes (folklore)	$\Theta(\kappa)$ by constant stepsizes (folklore)
Optimized stepsizes	$\Theta(\sqrt{\kappa})$ by Chebyshev Stepsizes [49]	$O(\kappa^{\log_\rho 2})$ by Silver Stepsizes [3] (Theorem 2)
Additional dynamics	$\Theta(\sqrt{\kappa})$ by Heavy Ball [33]	$\Theta(\sqrt{\kappa})$ by Nesterov Acceleration [31]

Table 1: Iteration complexity of various approaches for minimizing a κ-conditioned convex function. The dependence on the accuracy ε is omitted as it is always $\log 1/\varepsilon$. The story is analogous without strong convexity: rates of the form $O(\kappa^a \log 1/\varepsilon)$ here correspond to rates of the form $O(\varepsilon^{-a})$ there. This note focuses on a foundational question: how much mileage can one obtain from optimizing the stepsizes of GD?

$n = 2, 3$ in the strongly convex setting. Daccache [15] and Eloi [19] exhaustively extended these $n = 2, 3$ results to related settings and performance metrics. An exciting line of subsequent work numerically optimized longer schedules. Hedging over larger horizons n further improves the convergence rate, but searching for optimal stepsizes is a non-convex problem that is computationally difficult as n increases. Das Gupta et al. [16] developed a branch-and-bound framework to compute good schedules up to $n = 50$ for smooth convex optimization. Grimmer [20] developed a technique to round these branch-and-bound solutions to exact rational certificates; this allowed him to extend these approximate stepsize schedules up to $n = 127$ in order to get a larger constant-factor improvement.

How far can this be pushed? As n increases (beyond constants), can this time-varying hedging lead to *asymptotic* acceleration? What are the optimal stepsizes? Intriguing fractal-like patterns were observed for small n; do these qualitative behaviors persist for large n?

The two papers [3, 4] highlighted in this note answered this asymptotic-acceleration question affirmatively by introducing the *silver stepsize schedule*. This schedule is built recursively from the same short-step/long-step splitting pattern that appears in the optimal two-step schedule of [2]. In the smooth convex setting (Theorem 1), the silver stepsizes improve the iteration complexity of GD from the textbook rate $O(\varepsilon^{-1})$ to $O(\varepsilon^{-\log_\rho 2}) \approx O(\varepsilon^{-0.78})$. In the m-strongly convex and M-smooth setting (Theorem 2), they similarly improve the dependence on the condition number $\kappa = \frac{M}{m}$ from $O(\kappa \log \frac{1}{\varepsilon})$ for constant stepsizes to $O(\kappa^{\log_\rho 2} \log \frac{1}{\varepsilon}) \approx O(\kappa^{0.78} \log \frac{1}{\varepsilon})$. These asymptotic rates were conjectured to be asymptotically optimal among all non-adaptive stepsize schedules [3, 4]. (Concurrent work by Grimmer et al. [21] proved asymptotic acceleration using a different stepsize schedule, but at a suboptimal convergence rate, with exponent ≈ 0.95 rather than $\log_\rho 2 \approx 0.78$.)

Excitingly, in only the few years since, many papers have followed up; see §5 for a discussion of this burgeoning area.

2 Optimal stepsizes for quadratic optimization (Young 1953)

As mentioned above, in the special case of *quadratic* optimization, it is classically known that time-varying stepsizes can accelerate GD. This result is due to Young [49] and is nowadays taught in introductory optimization courses. We briefly recall this elegant argument below as it provides perhaps the simplest example of the broader algorithmic opportunity.

Young's stepsizes. Consider running GD on the class $\mathcal{F}$ of quadratic functions f that are m-strongly convex and M-smooth. What stepsize schedules make GD converge

$$\|x_n - x^*\| \le R_n \|x_0 - x^*\| \tag{4}$$

at the fastest possible rate R_n? Without loss of generality after translating, $f(x) = \frac{1}{2} x^T H x$ where $mI \preceq H \preceq MI$. Since f is quadratic, its gradient is linear $\nabla f(x) = Hx$, hence GD is a linear map $x_{t+1} = x_t - \alpha_t \nabla f(x_t) = (I - \alpha_t H) x_t$, and therefore the n-th iterate is

$$x_n = p_n(H) x_0 \,, \qquad \text{where} \qquad p_n(H) = \prod_{t<n} (I - \alpha_t H). \tag{5}$$

Observe that as one ranges over all possible choices of the stepsize schedule $\{\alpha_t\}_{t<n}$, the polynomial p_n ranges over the set $\mathcal{P}_n$ of all degree-n polynomials satisfying the normalizing condition $p_n(0) = 1$. Therefore finding an optimal stepsize schedule is equivalent to finding an optimal polynomial $p_n \in \mathcal{P}_n$.

What is the optimal polynomial? By the above display and basic properties of the spectral norm,

$$R_n = \sup_{f\in\mathcal{F},\, x_0\neq x^*} \frac{\|x_n - x^*\|}{\|x_0 - x^*\|} = \sup_{mI\preceq H\preceq MI,\, x_0\neq 0} \frac{\|p_n(H)x_0\|}{\|x_0\|} = \sup_{mI\preceq H\preceq MI} \|p_n(H)\| = \sup_{m\leq\lambda\leq M} |p_n(\lambda)|\,.$$

Thus the optimal polynomial $p_n \in \mathcal{P}_n$ is the one with minimal L_∞ norm over the interval $[m, M]$. It is classically known that this is the (translated and scaled) Chebyshev polynomial of the first kind, see e.g., [35]. By the definition of p_n in (5), it follows that the optimal stepsizes $\{\alpha_t\}_{t<n}$ are the inverses of the roots of this Chebyshev polynomial, in any order. See Figure 1 for an illustration and an interpretation of this phenomenon through the lens of *hedging*.

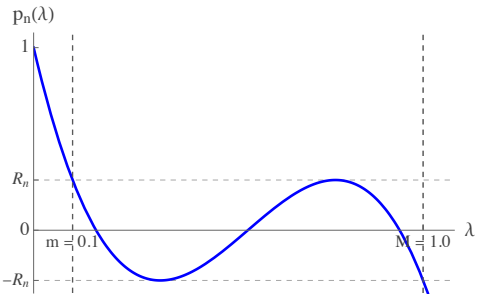


Figure 1: The translated and scaled Chebyshev polynomial $p_n \in \mathcal{P}_n$ has minimal L_∞ norm on $[m, M]$. Here visualized for $n = 4$, $m = 0.1$, $M = 1$. In quadratic optimization, the error after GD with stepsizes $\{\alpha_t\}_{t<n}$ is governed by the polynomial $p_n(\lambda) = \prod_{t<n}(1-\alpha_t\lambda)$. Chebyshev stepsizes hedge across curvatures $\lambda \in [m, M]$ by making the worst-case errors $p_n(\lambda)$ equioscillate between $\pm R_n$.

Beyond quadratics? This argument establishes that time-varying stepsizes can accelerate GD for quadratic f, but does this algorithmic opportunity extend beyond? From an analysis perspective, the argument fails from the very beginning: if f is not quadratic, then GD is not a linear map, and the equivalence to polynomials fails. In fact, this issue is not merely an artifact of analysis techniques: fundamental phenomena for the quadratic setting are false beyond. For example, in (non-quadratic) convex optimization, these Chebyshev stepsizes lead to divergence, the stepsize order matters, stepsizes and momentum are not equivalent, etc. As a result, it was unknown if *any* stepsize schedule could lead to *any* improvement over the constant stepsize schedule in the general setting of (non-quadratic) convex optimization. Is this a missed opportunity?

3 Optimal stepsizes for convex optimization, $n = 2$ (Altschuler 2018)

The answer is yes. This was first shown in [2]. In fact, that thesis showed several such results; we state here the simplest one.

Consider $n = 2$ steps, the minimal setting where time-varying stepsizes could possibly be advantageous. What two stepsizes α_0, α_1 make GD converge

$$\|x_2 - x^*\| \leq R_2\|x_0 - x^*\| \tag{6}$$

at the fastest possible rate R_2? In the worst case over objectives f that are m-strongly convex and M-smooth? Obviously $R_2 \leq R_1^2$ is possible by using $\alpha_0 = \alpha_1 = \bar{\alpha}$. Can strictly faster convergence $R_2 < R_1^2$ be achieved? With time-varying stepsizes $\alpha_0 \neq \alpha_1$? Remarkably, the answer is yes:

Theorem 3 (Theorem 8.10 of [2]). *The stepsizes α_0, α_1 that make R_2 as small as possible in (6) are unique, time-varying ($\alpha_0 \neq \alpha_1$), and strictly improve over the constant stepsize schedule ($R_2 < R_1^2$).*

As a trivial corollary, cyclically repeating $(\alpha_0, \alpha_1, \alpha_0, \alpha_1, \alpha_0, \alpha_1, \dots)$ strictly improves over the textbook rate for constant stepsizes. Indeed, the two-step convergence (6) implies $\|x_{2n} - x^*\| \leq R_2^n\|x_0 - x^*\|$, which strictly improves over the standard rate $\|x_{2n} - x^*\| \leq R_1^{2n}\|x_0 - x^*\|$ for constant stepsizes. See Figure 2.

The precise values for α_0, α_1, R_2 are complicated and not essential for this brief exposition[3]. We only highlight two features of these explicit values that will be built upon later:

[3]For the interested reader, the optimal stepsizes are $\alpha_0 = \frac{2}{m+S}$ and $\alpha_1 = \frac{2}{2M+m-S}$, with corresponding rate $R_2 = \frac{S-M}{2m+S-M}$, where $S = \sqrt{M^2 + (M-m)^2}$.

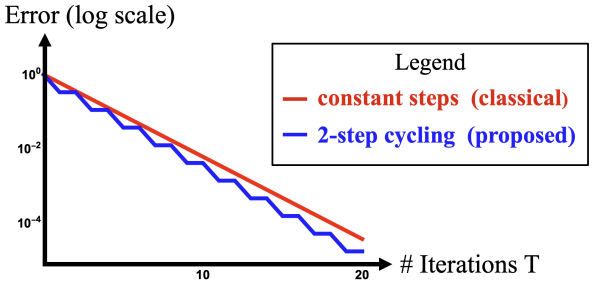


Figure 2: Illustration of Theorem 3: time-varying stepsizes that cycle between 2 stepsizes provably speed up GD for (non-quadratic) convex optimization. Here consider minimizing functions that are 1-strongly convex and 4-smooth. **Red:** the standard prescription of constant stepsizes $\alpha_t = 2/5$ in each iteration t leads to a convergence rate of 0.6^T after T iterations. **Blue:** better is *cycling* between $\alpha_t = 1/3$ and $1/2$ in even and odd steps; this leads to a faster convergence rate of $(1/\sqrt{3})^T \approx 0.57^T$.

- **Stepsize splitting.** The optimal stepsizes $\alpha_0 < \bar{\alpha} < \alpha_1$ are obtained by "splitting" the constant stepsize $\bar{\alpha}$ into a shorter step α_0 and a longer step α_1 as the two roots of a certain explicit quadratic equation.
- **Silver ratio.** The improvement $R_2 < R_1^2$ can be quantified as $R_2 \approx 1 - \frac{2(1+\sqrt{2})}{\kappa}$ whereas $R_1^2 \approx 1 - \frac{4}{\kappa}$, in the relevant asymptotic regime as $\kappa \to \infty$. This quantity $\rho = 1 + \sqrt{2}$ is called the silver ratio.

Summarizing, Theorem 3 shows a missed algorithmic opportunity: time-varying stepsizes provably make GD converge faster. However, this result only shows a constant factor improvement in the final iteration complexity (from $R_2 < R_1^2$) since it only considers $n = 2$ steps. The improvement increases in n. But by how much? As $n \to \infty$, what are the optimal stepsizes and rate? How much faster can one accelerate GD?

4 Silver stepsizes for convex optimization, $n \to \infty$ (Altschuler-Parrilo 2023)

The papers [3, 4] developed the *silver stepsizes* in order to prove acceleration for arbitrarily large horizons $n \to \infty$. The silver stepsizes accelerate the iteration complexity of GD from the textbook rate $O(\varepsilon^{-1})$ to $O(\varepsilon^{-\log_\rho 2})$ in the convex setting (Theorem 1), and analogously from $O(\kappa \log \frac{1}{\varepsilon})$ to $O(\kappa^{\log_\rho 2} \log \frac{1}{\varepsilon})$ in the strongly convex setting (Theorem 2). Here $\rho = 1 + \sqrt{2}$ denotes the silver ratio. In these papers we conjectured that our asymptotic rate is optimal among (non-adaptive) stepsize schedules.[4]

The silver stepsizes are constructed in a recursive manner from the 2-step solution in Theorem 3, see Figure 3. This leads to highly unconventional features: the silver stepsize schedule is non-monotonically time-varying, fractal-like, and has peaks that are exponentially infrequent but exponentially large. In the strongly-convex setting, the silver stepsizes are also approximately periodic with period of size $\kappa^{\log_\rho 2}$. See [3] for details.

The silver stepsize schedule simplifies in the (non-strongly) convex setting. See Figure 4. This is the formal limit as the strong convexity parameter $m \to 0$, or equivalently $\kappa \to \infty$. The resulting schedule has simple direct definitions, both recursively and explicitly. For simplicity, below we normalize the smoothness $M = 1$ (without loss of generality since running GD on f amounts to rescaling the stepsizes by $1/M$).

- **Recursive definition.** For any integer $n = 2^k - 1$, we recursively construct the schedule h_{2n+1} of length $2n + 1$ from the schedule h_n of length n via

$$h_{2n+1} = [h_n,\ 1 + \rho^{k-1},\ h_n] \tag{7}$$

  with base case $h_1 = [\sqrt{2}]$. This results in the pattern $[\sqrt{2},\ 2,\ \sqrt{2},\ 1 + \sqrt{2}, \dots]$ depicted in Figure 4.
- **Explicit definition.** The t-th stepsize is

$$\alpha_t = 1 + \rho^{\nu(t+1)-1} \tag{8}$$

[4] If one believes the optimality of our asymptotic rates, then one can ask even more fine-grained questions about the hidden constant in the big-O notation. We further conjecture that the silver stepsize schedule is exactly optimal in the strongly convex setting for the performance metric $\|x_n - x^*\|/\|x_0 - x^*\|$ that it was originally designed for. In the convex setting, an interesting line of follow-up work has improved the hidden-constant by almost a factor of 3 and presented conjecturally optimal stepsizes for a variety of performance metrics, see the discussion in §5.

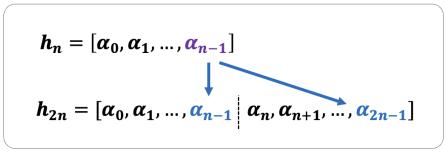


Figure 3: Schematic of how the silver stepsize schedule is recursively constructed in the strongly convex setting [3]. Let h_n denote the schedule of length n. Then h_{2n} is built from two copies of h_n, but with a key modification of the final step α_{n-1} in h_n (purple). It is "split" into a shorter step α_{n-1} and longer step α_{2n-1} (blue), given by the two roots of a quadratic equation, exactly analogous to the 2-step construction in §3.

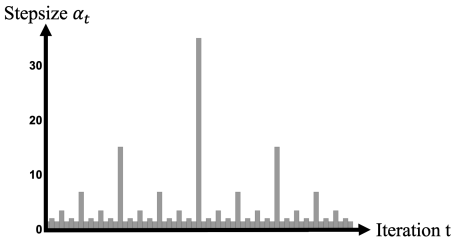


Figure 4: The silver stepsize schedule we proposed for convex optimization [3, 4]. The first 63 stepsizes are shown. They are highly unconventional: time-varying, non-monotonic, fractal-like, and sometimes use extremely aggressive stepsize "spikes" of size $\alpha_{2^k-1} = 1 + \rho^{k-1}$ where $\rho = 1 + \sqrt{2}$ denotes the silver ratio.

where $\nu(t)$ denotes the 2-adic valuation of t, i.e., the largest non-negative integer i such that 2^i divides t. For example, $\nu(1) = 0$, $\nu(2) = 1$, $\nu(3) = 0$, $\nu(4) = 2$, etc.

Algorithm analysis: multi-step descent. The core technical challenge is that hedging requires deviating from standard analyses, which bound the progress of each step individually (e.g., using the descent lemma) and then sum the progress. Such *one-step* analyses are too shortsighted to prove acceleration. Instead, one must directly analyze the *multi-step progress* in order to show that different iterations of GD synergize with each other. These holistic analyses are necessary for proving *any* benefit of time-varying stepsizes.

Of course, directly analyzing the long-term progress is much more difficult. This involves answering two interrelated questions, both of which are non-trivial:

- *Analysis question:* how to analyze a given stepsize schedule?
- *Design question:* how to design the optimal stepsize schedule?

Our starting point is the performance estimation problem (PEP) framework, pioneered by [17] and refined by [43], which enables numerically solving the analysis question using semidefinite programming (SDP). PEP addresses the analysis question by linearly combining valid inequalities for the class of functions under consideration; in the convex case, these are the cocoercivities relating function values and gradients at different points. Although PEP is helpful, it is important to emphasize that it is *not* a complete solution to the problem at hand. One challenge is that PEP does not fully solve the analysis question: it only produces a numerical estimate of the convergence rate for a fixed number of iterations n, whereas establishing acceleration requires rigorous symbolic proofs for arbitrarily large n. The biggest limitation, however, is that PEP does not directly tackle the design question: finding the stepsize schedule with fastest convergence is non-convex in all existing PEP formulations[5]. This poses a key difficulty for discovering hedging strategies.

Our work [3, 4] proposes a technique called *recursive gluing* which helps make both the design and analysis questions tractable for arbitrarily large n. In terms of the design question, as described above, we recursively construct the silver stepsize schedule of size $2n$ by gluing together two copies of the schedule of length n, modulo modification to a constant number of stepsizes. A key insight is that we can leverage this recursive structure of the stepsize schedule in order to also recursively approach the analysis question: we show that one can "recursively glue" two copies of the convergence proof for the smaller schedules, modulo modification to a constant number of inequalities. The key point is that the proof complexity does not increase in n. Indeed, in this way, proving the $2n$-step convergence rate from the n-step rate is no harder than proving the 2-step convergence rate from the textbook 1-step rate. See [3, 4] for full details.

[5] It is an interesting question if there are alternative formulations or parameterizations of the stepsize design question that are convex.

5 Outlook

This general principle of multi-step descent opens up many directions for the design and analysis of algorithms, as it suggests a potential missed opportunity for any optimization algorithm analyzed with traditional one-step analyses. In just the past three years since [3, 4], many papers have already followed up. For example, recent work has showcased the generality of this stepsize-hedging phenomenon by extending the results to anytime convergence [25, 52], constrained and proximal settings [8, 9, 46], Riemannian settings [32], robustness and inexact gradient settings [44, 45**?**], operator-splitting settings [1], min-max settings [39, 40], random stepsizes [5], and other performance metrics [22, 23, 51]. The line of work [22, 23, 46, 51] has also improved the hidden constant in the big-O for the non-strongly convex setting (although still with the same asymptotic rate $\log_\rho 2$ as the silver stepsize schedule), and in particular [23, 51] developed composition/concatenation analysis frameworks which simplify and refine the recursive gluing technique of [3, 4] and suggest stepsize schedules that are conjecturally minimax-optimal for several performance metrics. Recent work has also used large stepsizes to obtain faster rates or sharper dynamical understanding for logistic/classification losses, including separable logistic regression, regularized logistic regression, non-separable settings, and related two-layer-network models [6, 12, 13, 14, 27, 28, 47, 48, 50]. The machine learning community has also been excited by intriguing similarities with similar stepsize schedules previously explored in empirical ML, e.g., the cyclical stepsize schedules of [41, 42] as well as learned schedules in parametric optimization [36, 37, 38]. Previously, time-varying stepsizes had no provable benefit even in (non-quadratic) convex optimization; this line of work provides a first step towards explaining the unreasonable effectiveness of time-varying stepsizes in empirical deep learning. But much more is needed to realize the potential of time-varying hedging as a powerful algorithmic primitive in both theory and practice.

References

[1] Abbaszadehpeivasti H, Zamani M (2025) On the convergence rate of the Douglas-Rachford splitting algorithm. *Preprint at arXiv:2509.06676* .

[2] Altschuler JM (2018) *Greed, Hedging, and Acceleration in Convex Optimization*. Master's thesis, Massachusetts Institute of Technology.

[3] Altschuler JM, Parrilo PA (2025) Acceleration by Stepsize Hedging: Multi-Step Descent and the Silver Stepsize Schedule. *Journal of the ACM* 72(2):1–38.

[4] Altschuler JM, Parrilo PA (2025) Acceleration by stepsize hedging: Silver Stepsize Schedule for smooth convex optimization. *Mathematical Programming* 213(1–2):1105–1118.

[5] Altschuler JM, Parrilo PA (2026) Acceleration by random stepsizes: Hedging, equalization, and the arcsine stepsize schedule. *Foundations of Computational Mathematics,* to appear.

[6] Axiotis K, Sviridenko M (2023) Gradient descent converges linearly for logistic regression on separable data. *International Conference on Machine Learning*, 1302–1319.

[7] Bertsekas DP (1999) *Nonlinear programming* (Athena Scientific).

[8] Bok J, Altschuler JM (2025) Accelerating proximal gradient descent via silver stepsizes. *Conference on Learning Theory*, 421–453.

[9] Bok J, Altschuler JM (2026) Optimized methods for composite optimization: a reduction perspective. *Mathematical Programming,* to appear.

[10] Boyd S, Vandenberghe L (2004) *Convex optimization* (Cambridge University Press).

[11] Bubeck S (2015) Convex optimization: Algorithms and complexity. *Foundations and Trends in Machine Learning* 8(3-4):231–357.

[12] Cai Y, Wu J, Mei S, Lindsey M, Bartlett PL (2024) Large stepsize gradient descent for non-homogeneous two-layer networks: Margin improvement and fast optimization. *Advances in Neural Information Processing Systems*.

[13] Crawshaw M, Liu M (2026) Tight bounds for logistic regression with large stepsize gradient descent in low dimension. *Preprint at arXiv:2602.12471* .

[14] Crawshaw M, Woodworth B, Liu M (2025) Constant stepsize local GD for logistic regression: Acceleration by instability. *Preprint at arXiv:2506.13974* .

[15] Daccache A (2019) *Performance estimation of the gradient method with fixed arbitrary step sizes*. Master's thesis, Université Catholique de Louvain.

[16] Das Gupta S, Van Parys BP, Ryu EK (2024) Branch-and-bound performance estimation programming: a unified methodology for constructing optimal optimization methods. *Mathematical Programming* 567—-639.

[17] Drori Y, Teboulle M (2014) Performance of first-order methods for smooth convex minimization: a novel approach. *Mathematical Programming* 145(1–2):451–482.

[18] d'Aspremont A, Scieur D, Taylor A (2021) Acceleration methods. *Foundations and Trends® in Optimization* 5(1-2):1–245.

[19] Eloi D (2022) *Worst-case functions for the gradient method with fixed variable step sizes*. Master's thesis, Université Catholique de Louvain.

[20] Grimmer B (2024) Provably faster gradient descent via long steps. *SIAM Journal on Optimization* 34(3):2588–2608.

[21] Grimmer B, Shu K, Wang AL (2023) Accelerated gradient descent via long steps. *Preprint at arXiv:2309.09961* .

[22] Grimmer B, Shu K, Wang AL (2025) Accelerated objective gap and gradient norm convergence for gradient descent via long steps. *INFORMS Journal on Optimization* 7(2):156–169.

[23] Grimmer B, Shu K, Wang AL (2026) Composing optimized stepsize schedules for gradient descent. *Mathematics of Operations Research,* to appear.

[24] Hazan E (2016) Introduction to online convex optimization. *Foundations and Trends in Optimization* 2(3-4):157–325.

[25] Kornowski G, Shamir O (2024) Open problem: Anytime convergence rate of gradient descent. *Conference on Learning Theory*, volume 247.

[26] Luenberger DG, Ye Y (1984) *Linear and nonlinear programming* (Springer).

[27] Meng SY, Goujaud B, Orvieto A, De Sa C (2025) Gradient descent on logistic regression: Do large step-sizes work with data on the sphere? *Preprint at arXiv:2507.11228* .

[28] Meng SY, Orvieto A, Cao DY, De Sa C (2024) Gradient descent on logistic regression with non-separable data and large step sizes. *Preprint at arXiv:2406.05033* .

[29] Nemirovskii A, Yudin DB (1983) *Problem complexity and method efficiency in optimization* (Wiley).

[30] Nesterov Y (2013) *Introductory lectures on convex optimization: A basic course*, volume 87 (Springer Science & Business Media).

[31] Nesterov YE (1983) A method of solving a convex programming problem with convergence rate $O(1/k^2)$. *Soviet Math. Dokl.* 27(2):372–376.

[32] Park J, Roy A, Siegel JW, Bhattacharya A (2025) Acceleration via silver step-size on Riemannian manifolds with applications to Wasserstein space. *Advances in Neural Information Processing Systems* 38.

[33] Polyak BT (1964) Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics* 4(5):1–17.

[34] Polyak BT (1987) *Introduction to optimization* (Optimization Software, Inc.).

[35] Rivlin TJ (1981) *An introduction to the approximation of functions* (Courier Corporation).

[36] Sambharya R, Bok J, Matni N, Pappas G (2025) Learning acceleration algorithms for fast parametric convex optimization with certified robustness. *Preprint at arXiv:2507.16264* .

[37] Sambharya R, Stellato B (2025) Data-driven performance guarantees for classical and learned optimizers. *Journal of Machine Learning Research* 26(171):1–49.

[38] Sambharya R, Stellato B (2026) Learning algorithm hyperparameters for fast parametric convex optimization. *SIAM Journal on Mathematics of Data Science,* to appear.

[39] Shugart H, Altschuler JM (2025) Min-max optimization is strictly easier than variational inequalities. *Preprint at arXiv:2511.03052* .

[40] Shugart H, Altschuler JM (2025) Negative stepsizes make gradient-descent-ascent converge. *Preprint at arXiv:2505.01423* .

[41] Smith LN (2017) Cyclical learning rates for training neural networks. *IEEE Winter Conference on Applications of Computer Vision*, 464–472 (IEEE).

[42] Smith LN, Topin N (2019) Super-convergence: Very fast training of neural networks using large learning rates. *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, volume 11006, 369–386 (SPIE).

[43] Taylor AB, Hendrickx JM, Glineur F (2017) Smooth strongly convex interpolation and exact worst-case performance of first-order methods. *Mathematical Programming* 161(1-2):307–345.

[44] Vernimmen P (2024) *Tight convergence analysis of exact and inexact gradient methods with constant and silver schedules*. Master's thesis, Université Catholique de Louvain.

[45] Vernimmen P, Glineur F (2025) Empirical and computer-aided robustness analysis of long-step and accelerated methods in smooth convex optimization. *International Conference on Modelling, Computation and Optimization in Information Systems and Management Sciences*, 354–366 (Springer).

[46] Wang B, Ma S, Yang J, Zhou D (2026) Relaxed proximal point algorithm: Tight complexity bounds and acceleration without momentum. *INFORMS Journal on Optimization* 8(2):141–162.

[47] Wu J, Bartlett PL, Telgarsky M, Yu B (2024) Large stepsize gradient descent for logistic loss: Non-monotonicity of the loss improves optimization efficiency. *Proceedings of the Thirty Seventh Conference on Learning Theory*, Proceedings of Machine Learning Research (PMLR).

[48] Wu J, Marion P, Bartlett P (2025) Large stepsizes accelerate gradient descent for regularized logistic regression. *Advances in Neural Information Processing Systems* 38:104485–104525.

[49] Young D (1953) On Richardson's method for solving linear systems with positive definite matrices. *Journal of Mathematics and Physics* 32(1-4):243–255.

[50] Zhang R, Wu J, Bartlett PL (2025) Gradient descent converges arbitrarily fast for logistic regression via large and adaptive stepsizes. *International Conference on Machine Learning*, volume 267, 76361–76384.

[51] Zhang Z, Jiang R (2026) Accelerated gradient descent by concatenation of stepsize schedules. *SIAM Journal on Optimization,* to appear.

[52] Zhang Z, Lee J, Du S, Chen Y (2025) Anytime acceleration of gradient descent. *Conference on Learning Theory*, volume 291, 5991–6013.

Research Highlight
Robust Paths: Geometry and Computation

by HAO HAO[1], PETER ZHANG[1]
[1] HEINZ COLLEGE OF INFORMATION SYSTEMS AND PUBLIC POLICY, CARNEGIE MELLON UNIVERSITY
HAOHAO@ANDREW.CMU.EDU, PYZHANG@CMU.EDU

We would like to sincerely thank the ICS Student Paper Award Committee: Ryan Cory-Wright, Austin Buchanan (Chair), Yongchun Li, and Young Woong Park for recognizing our work with the honor. We would also like to thank the ICS Newsletter Editor, Hamed Rahimian, for kindly inviting us to share this work with the community. This article is a summary of our work in [11].

1 Introduction

Robust optimization [3] is a method for addressing uncertainty in optimization problems, with an extensive modeling capability across problem classes and uncertainty structures [1, 7, 18, 19]. In this work, we study robust optimization problems with uncertain objective functions:

$$\text{(RC)} \quad \min_{x\in\mathcal{X}} \max_{a\in\mathcal{U}} \langle a, x\rangle, \tag{1}$$

where decision x belongs to a closed, convex and nonempty feasible region $\mathcal{X} \subseteq \mathbb{R}^n$, a is a vector of uncertain parameters and is only known to reside in an uncertainty set: $\mathcal{U} = \{a_0+\xi : \xi \in \Xi \subset \mathbb{R}^n\}$. Here a_0 is the nominal vector, ξ is the uncertain perturbation assumed to be in a compact and convex set Ξ. We consider general uncertainty sets representable via gauge function [8, 9, 16] constraints: $\Xi(r, \mathcal{V}) = \{\xi \in \mathbb{R}^n : \|\xi\|_{\mathcal{V}} \leq r\}$, where the gauge function is defined as $\|v\|_{\mathcal{V}} = \inf\{t \geq 0 : v \in t\mathcal{V}\}$, and the size and shape of $\Xi(r, \mathcal{V})$ can be flexibly adjusted via radius r and gauge set $\mathcal{V}$ respectively. We assume $r \geq 0$ and $\mathcal{V}$ is a compact and convex set with $0 \in \text{int}(\mathcal{V})$.

A major challenge when deploying robust optimization in real-world problems is the calibration of the uncertainty set parameterized by its size and shape $(r, \mathcal{V})$, a choice that directly affects the efficiency and the robustness trade-off of the robust decisions. Existing approaches calibrate $(r, \mathcal{V})$ based on probabilistic guarantees [4, 5, 6, 13]. However, such approaches assume prior knowledge of the uncertainty distribution or observations on the uncertainty, which can be unavailable in practice. Even with distributional information or data on the uncertainty, it has been observed that the resulting robust solutions from this approach can be too conservative [15]. Often, the practical approach is costly and ad hoc: solving the robust optimization problem multiple times, each under a different choice of $(r, \mathcal{V})$, before comparing the performance of the different robust solutions (which includes cross-validation among other statistical methods for hyperparameter tuning) [2, 13, 15]. To this end, ideally, decision makers need *the entire set of robust solutions* under multiple shapes $\mathcal{V}$ and radii r.

Definition 1 (Robust Path)**.** *The* robust path *of* (RC) *under* $\mathcal{V}$ *is defined as*

$$\mathcal{P}(\mathcal{V}) = \left\{ x_{\text{R}}(r, \mathcal{V}) \in \arg\min_{x\in\mathcal{X}} \max_{\xi\in\Xi(r,\mathcal{V})} \langle a_0 + \xi, x\rangle : r \in [0, \infty) \right\}.$$

The challenge lies in obtaining the robust paths, potentially without repeatedly solving the robust optimization problem. To this end, we investigate the following questions: 1. What are the structures of robust paths? 2. Given the structural information, can we find algorithms to trace the robust paths?

Contribution. In this paper, we answer both questions positively:

1. We characterize the geometry of the robust paths as the Bregman projection of curves (whose geometries are defined by the uncertainty set shape $\mathcal{V}$) onto the feasible region $\mathcal{X}$.
2. Once the appropriate geometric lens is established, we find a surprisingly simple way to approximate robust paths. We connect the following two *conceptually distinct*, yet *geometrically similar* solution sets: a) the robust paths of robust optimization problem (RC) and b) the optimization paths of the deterministic optimization counterpart. Specifically: **2.1.** The proximal point method (PPM) optimization paths for solving the deterministic counterpart initialized at the "most robust" solution are close approximate, sometimes even exact, robust paths. **2.2.** The design of the robust path uncertainty set shape $\mathcal{V}$ is equivalent to the choice of the PPM distance-generating function; adjusting the cadence of the robust solutions' radii r, corresponds exactly to adjusting the step-size of the PPM. **2.3.** The distance between the PPM approximation and the exact robust path hinges precisely on the geometry of the feasible region $\mathcal{X}$ and the uncertainty set shape $\mathcal{V}$.

2 A Unified Geometric View of Robust Path and Optimization Paths

We make the following assumptions throughout the remainder of the article.

Assumption 1. *The uncertainty set $\mathcal{V}$ is compact, smooth, and strictly convex, with $0 \in \operatorname{int}(\mathcal{V})$.*

Assumption 2. *Define $\varphi(\cdot) = g \circ \|\cdot\|_{\mathcal{V}^\circ}$, where $g : \mathbb{R} \to \mathbb{R}_+$ is Legendre, $g(0) = 0$ and $\nabla g(0) = 0$.*

Note that the two are technical assumptions to facilitate analysis, not a restriction on (RC). For instance, for polytope uncertainty sets, there exist simple, infinitesimally close approximations that satisfy Assumption 1.

In Definition 1, we defined a robust path as the set of (potentially non-unique) optimal solutions of (RC) with different radii. Now with the help of Assumptions 1 and 2, we refine Definition 1 into Definition 2, where each radius r corresponds to a *unique* optimal solution. The main implication of the solution uniqueness of $\mathcal{P}'(\mathcal{V})$ is that its geometry can be precisely characterized via Bregman projection.

Definition 2 ((Characterizable) Robust Path)**.** *The* (characterizable) robust path *of* (RC) *under $\mathcal{V}$ is defined as $\mathcal{P}'(\mathcal{V}) = \{x'_{\mathrm{R}}(\omega, \mathcal{V}) : \omega \in [0, \infty)\}$, where*

$$x'_{\mathrm{R}}(\omega) := x'_{\mathrm{R}}(\omega, \mathcal{V}) = \arg\min_{x \in \mathcal{X}} \langle a_0, x\rangle + \omega\varphi(x) = \arg\min_{x \in \mathcal{X}} \langle a_0, x\rangle + \omega \cdot g \circ \|x\|_{\mathcal{V}^\circ}.$$

The robust solution x_{R} and the efficient solution x_{E} are defined as $x_{\mathrm{R}} = \lim_{\omega\to\infty} x'_{\mathrm{R}}(\omega, \mathcal{V}) = \arg\min_{x\in\mathcal{X}} \varphi(x)$ and $x_{\mathrm{E}} = \lim_{\omega\to 0} x'_{\mathrm{R}}(\omega, \mathcal{V}) = \arg\min_{x\in\mathcal{X}} \langle a_0, x\rangle$.

Next, we define two optimization paths for solving the nominal problem

$$(\mathrm{P}) \quad \min_{x \in \mathcal{X}} \langle a_0, x\rangle, \tag{2}$$

where the parameter a is assumed to be deterministically its nominal value a_0.

Definition 3 (Proximal Path)**.** *Given a distance-generating function ψ, a step-size sequence $\{\lambda_k > 0\}$, and a starting point $x_0 \in \mathcal{X}$, the* proximal path *for solving problem* (P) *is the sequence $\{x_k\}$ defined as*

$$x_{k+1} = \arg\min_{x \in \mathcal{X}} \langle a_0, x\rangle + \lambda_k D_\psi(x, x_k),\ k = 0, 1, \ldots$$

Definition 4 (Central Path)**.** *The* central path *for solving problem* (P)*, induced by distance-generating function ψ and initialized at x_0 is the set $\{x_{\mathrm{CP}}(\omega) : \omega \in [0, \infty)\}$ defined as*

$$x_{\mathrm{CP}}(\omega) = \arg\min_{x \in \mathcal{X}} \langle a_0, x\rangle + \omega D_\psi(x, x_0).$$

Our first main result reveals that the robust path $\mathcal{P}'(\mathcal{V})$ of (RC) and two appropriately defined optimization paths of (P) are in fact geometrically similar under the lens of Bregman projection.

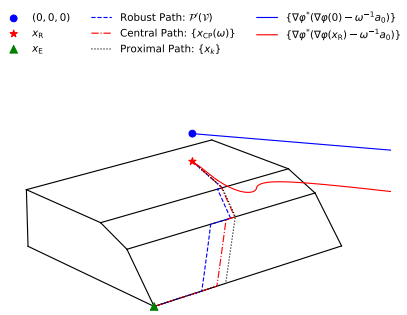
Theorem 4. *Denote $\mathcal{P}'(\mathcal{V})$ as the* robust path *of* (RC) *according to Definition 2. Define $x_{\mathrm{R}} = \lim_{\omega\to\infty} x'_{\mathrm{R}}(\omega, \mathcal{V}) = \arg\min_{x\in\mathcal{X}} \varphi(x)$. Denote $\{x_{\mathrm{CP}}(\omega) : \omega \in [0, \infty)\}$ as the* central path *for solving* (P)*, using $\varphi(\cdot) = g \circ \|\cdot\|_{\mathcal{V}^\circ}$ as the d.g.f. and initialized at x_{R}. Denote $\{x_k\}$ as the* proximal path *for solving* (P)*, using $\varphi(\cdot) = g \circ \|\cdot\|_{\mathcal{V}^\circ}$ as the d.g.f., under a step-size sequence $\{\lambda_k > 0 : \sum_{k=0}^{\infty} \lambda_k^{-1} = \infty\}$ and initialized at x_{R}. The three paths have the following Bregman projection interpretation:*

$$\begin{aligned}
&\text{(Robust Path)} && \mathcal{P}'(\mathcal{V}) = \left\{\Pi_{\mathcal{X}}^{\varphi}\left(\nabla\varphi^*\left(\nabla\varphi(\,0\,) - \omega^{-1}a_0\right)\right) : \omega \in [0,\infty)\right\},\\
&\text{(Central Path)} && \{x_{\mathrm{CP}}(\omega) : \omega \in [0,\infty)\} = \left\{\Pi_{\mathcal{X}}^{\varphi}\left(\nabla\varphi^*\left(\nabla\varphi(x_{\mathrm{R}}) - \omega^{-1}a_0\right)\right) : \omega \in [0,\infty)\right\},\\
&\text{(Proximal Path)} && x_{k+1} = \Pi_{\mathcal{X}}^{\varphi}\left(\nabla\varphi^*\left(\nabla\varphi(x_k) - \lambda_k^{-1}\,a_0\right)\right),\ \text{for } k = 0, 1, \cdots,\ x_0 = x_{\mathrm{R}}.
\end{aligned}$$

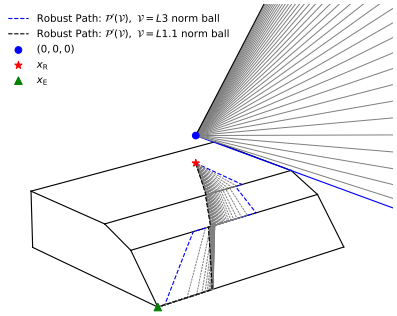
In Figure 1a, we visualize, as Theorem 4 entails, the Bregman projection interpretation of the three paths under a single $\mathcal{V}$. Figure 1b presents the geometry of a set of robust paths $\mathcal{P}'(\mathcal{V})$ under varying $\mathcal{V}$ designs.

3 Recovering Robust Path by Optimization Paths

In this section, leveraging the geometric view of Section 2, we show that the robust path $\mathcal{P}'(\mathcal{V})$ of (RC) can be approximated by some appropriately designed central path $\{x_{\mathrm{CP}}(\omega)\}$ and proximal path $\{x_k\}$ of (P).



(a) Geometry of the robust path $\mathcal{P}'(\mathcal{V})$, the central path $\{x_{\mathrm{CP}}(\omega)\}$, and the proximal path $\{x_k\}$.



(b) Geometry of a set of robust paths under varying uncertainty sets.

Figure 1: Theorem 4: Unified Bregman projection view of robust paths of (RC), central path and proximal path of (P).

3.1 Robust Path and Central Path

Theorem 5. *Assume $\mathcal{V}$ satisfies Assumption 1, φ satisfies Assumption 2, and the induced Bregman projection $\Pi^{\varphi}_{\mathcal{S}}$ is κ-expansive. The Bregman divergence between the central path and the robust path is uniformly bounded by an upper bound. The upper bound is sharp and depends on κ and the Bregman divergence between two points.*

$$D_{\varphi}\left(x_{\mathrm{CP}}(\omega), x'_{\mathrm{R}}(\omega)\right) \leq \kappa^2 \cdot D_{\varphi}\left(\Pi^{\varphi}_{\mathcal{X}}(0), \Pi^{\varphi}_{\mathrm{Aff}(\mathcal{X})}(0)\right), \quad \forall \omega \in [0, \infty).$$

3.2 Central Path and Proximal Path

In this section, we show the two optimization paths of (P), *i.e.*, the proximal path $\{x_k\}$ and the central path $\{x_{\mathrm{CP}}(\omega)\}$ are in general approximations of each other, and under two special cases equivalent.

3.2.1. Special Case One: Polyhedral Monotonicity of Feasible Regions. For convex polyhedron feasible regions $\mathcal{X}$, we show that if the following is true: once $\{x_k\}$ enters a face of $\mathcal{X}$ it remains in that face, then $\{x_k\} = \{x_{\mathrm{CP}}(\omega_k)\}$. More precisely, we adopt the following definition from [10].

Definition 5. *The proximal path $\{x_k\}$ is monotone on convex polyhedron $\mathcal{X}$, if for any face $\mathcal{F}$ of $\mathcal{X}$: $x_k \in \mathcal{F} \quad \Rightarrow \quad x_{k+n} \in \mathcal{F}$, $\forall n \in [1, K]$.*

Proposition 1. *Let the feasible region $\mathcal{X}$ be a convex polyhedron. Let $\{x_k\}$ be a proximal path initialized by x_0 associated with the step-size sequence $\{\lambda_k > 0 : \sum_{k=0}^{\infty} \lambda_k^{-1} = \infty\}$. Let $\{x_{\mathrm{CP}}(\omega_k)\}$ be the central path initialized also at x_0. If $\{x_k\}$ is monotone on $\mathcal{X}$, then $x_{k+1} = x_{\mathrm{CP}}(\omega_{k+1}), \forall k \in [0, K]$. Furthermore, ω_k can be recovered in closed form as a function of the proximal path step-size sequence $\{\lambda_k\}$: $\omega_k = \left(\lambda_0^{-1} + \cdots + \lambda_{k-1}^{-1}\right)^{-1}$.*

3.2.2. Special Case Two: Feasible Region and Uncertainty Set are Polar Pairs.

Proposition 2. *Assume $\mathcal{X} = \{x \in \mathbb{R}^n : \|x\|_{\mathcal{U}} \leq l\}$ where $\mathcal{U}$ satisfies Assumption 1, if robust path uncertainty set is designed to be $\mathcal{U}^{\circ}$ and the corresponding optimization path distance-generating function $\varphi(\cdot) = g \circ \|\cdot\|_{\mathcal{U}}$ satisfies Assumption 2, then: $\{x_k\} \subset \{x_{\mathrm{CP}}(\omega) : \omega \in [0, \infty)\}$.*

3.2.3. General Case. In general, the proximal path and the central path do not coincide. We establish a theoretical characterization of the distance between the two algorithmic paths with the following result.

Theorem 6. *Under Assumptions 1 and 2, denote $\{x_k\} = \{x_k : k \in [0, K]\}$ as the proximal path induced by φ whose Bregman projection is κ-expansive, initialized at x_0 and associated with a step-size sequence $\{\lambda_k > 0 : \sum_{k=0}^{\infty} \lambda_k^{-1} = \infty\}$. Partition the paths by the face of $\mathcal{X}$ they visit, $\mathcal{F}_i, \forall i \in [I]$. Then, for each $i \in [I]$ we have*

$$D_{\varphi}\left(x_k, x_{\mathrm{CP}}(\upsilon_k^{-1}; x_0)\right) \leq \kappa \cdot D_{\varphi}\left(x_{\mathrm{CP}}\left((\underline{\upsilon}^{(i)})^{-1}; x_0\right), x_{\underline{k}^{(i)}}\right), \quad \forall k \in \left[\underline{k}^{(i)} + 1, \overline{k}^{(i)}\right],$$

where $\upsilon_k = \underline{\upsilon}^{(i)} + \sum_{j=\underline{k}^{(i)}}^{k-1} \lambda_j^{-1}$.

3.3 Sufficient Condition: Proximal Paths are Exact Robust Paths

Finally, we close the loop and give a sufficient condition for the proximal path to be an exact robust path.

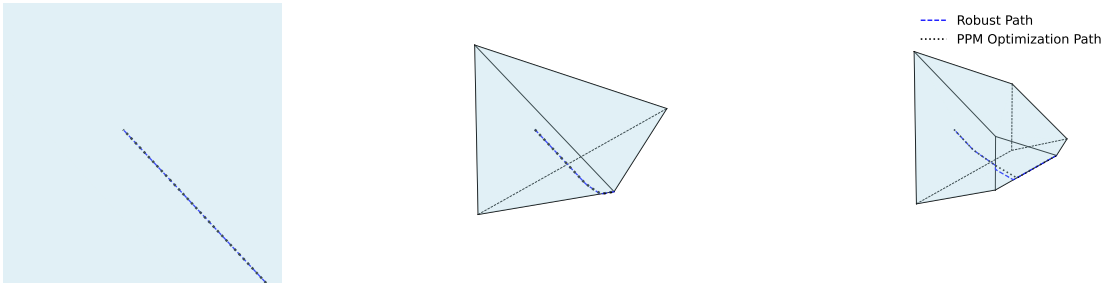
Theorem 7. *Let $\{x_k\}$ be the proximal path for (P) induced by $\varphi(\cdot) = g \circ \|\cdot\|_{\mathcal{V}^{\circ}}$, initialized at x_{R} and associated with $\{\lambda_k > 0 : \sum_{k=0}^{\infty} \lambda_k^{-1} = \infty\}$. Assume $\Pi^{\varphi}_{\mathcal{X}}(0) = \Pi^{\varphi}_{\mathrm{Aff}(\mathcal{X})}(0)$ and $\{x_k\}$ is monotone on $\mathcal{X}$, then for every k, x_k is a solution to (RC): $x_k \in \arg\min_{x \in X} \max_{\xi \in \Xi(r_k, \mathcal{V})} \langle a_0 + \xi, x \rangle$, where the corresponding uncertainty set radius r_k admits the following closed-form expression: $r_k = \omega_k \nabla g\left(\|x_k\|_{\mathcal{V}^{\circ}}\right)$, with $\omega_k = \left(\lambda_0^{-1} + \cdots + \lambda_{k-1}^{-1}\right)^{-1}$.*

4 Numerical Experiments

Portfolio Optimization. In portfolio optimization, we are concerned with constructing a portfolio from n risky assets. The return of the n assets is modeled by a random vector, α. We assume that from historical data, we can estimate the expectation and the covariance matrix of α to be μ and Σ. It is known that the classical Markowitz mean-variance portfolio optimization problem can be cast in the form of our (RC) with an ellipsoidal uncertainty set $\mathcal{U}(r) = \{\mu + \xi \in \mathbb{R}^n : \ \|\Sigma^{-1/2}\xi\|_2 \leq r\}$ [14]:

$$\min_{x \in \mathcal{X}} \max_{\alpha \in \mathcal{U}(r)} -\langle \alpha, x \rangle \tag{3}$$

Results. in Figure 2, we visualize in the solution space the robust paths of problem (3) (i.e., the set of Markowitz portfolios under varying mean-variance trade-off) and their proximal path approximations.



(a) Hyperplane. (b) Simplex. (c) General.

Figure 2: Robust path and its proximal path approximation of problem (3) in the solution space: The left two figures represent cases where our Theorem 7 predict a precise alignment between the proximal paths and robust paths. In the rightmost figure, our Theorem 6 predicts a small gap between the robust paths (blue line) and proximal paths (black line) since the proximal path is no longer monotone.

Adversarial training as robust optimization. The goal in adversarially robust deep learning is to learn networks that are robust against adversarial attacks (*i.e.*, perturbations on the input examples that aim to deteriorate the accuracy of classifiers). A common strategy to robustify networks is adversarial training, which can be formulated as the following robust optimization problem [12],

$$\min_{\theta} \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\max_{\xi \in \Xi(r,\mathcal{V})} \ell(f_\theta(x+\xi), y) \right], \tag{4}$$

Results. As the benchmark, we compute the robust paths of Problem (4) by solving (4) multiple times under varying levels of adversary attack budget r, with FGSM [17]. We then approximate the robust paths by simply performing standard training with approximate proximal point method (PPM), with the key choice of initializing the training at the most robust model weight. As shown in figure 3, the performance (clean accuracy v.s. adversarial accuracy) of our PPM approximated robust models (dashed curves) is comparable to the benchmark robust models (solid curve). At the same time, the computation cost of our method is 40 times lower than that of the FGSM benchmark (for generating 100 robust models with various levels of r).

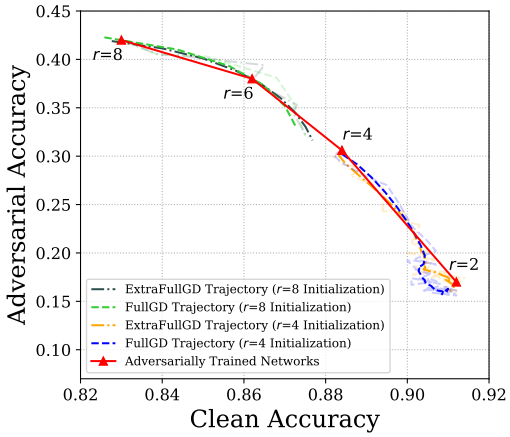


Figure 3: Robust Path of Problem (4): Our method v.s. FGSM.

References

[1] Alper Atamtürk and Muhong Zhang. Two-stage robust network flow and design under demand uncertainty. *Operations Research*, 55(4):662–673, 2007.

[2] Aharon Ben-Tal, Dick den Hertog, Anja De Waegenaere, Bertrand Melenberg, and Gijs Rennen. Robust solutions of optimization problems affected by uncertain probabilities. *Management Science*, 59(2):341–357, 2013.

[3] Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski. *Robust Optimization*. Princeton University Press, 1 edition, 2009.

[4] Dimitris Bertsimas, Dick Den Hertog, and Jean Pauphilet. Probabilistic guarantees in robust optimization. *SIAM Journal on Optimization*, 31(4):2893–2920, 2021.

[5] Dimitris Bertsimas and Melvyn Sim. The price of robustness. *Operations Research*, 52:35–53, 2 2004.

[6] Jose Blanchet, Yang Kang, Karthyek Murthy, and Fan Zhang. Data-driven optimal transport cost selection for distributionally robust optimization. In *2019 Winter Simulation Conference (WSC)*, pages 3740–3751, 2019.

[7] Erick Delage and Yinyu Ye. Distributionally robust optimization under moment uncertainty with application to data-driven problems. *Operations Research*, 58(3):595–612, 2010.

[8] Robert M Freund. Dual gauge programs, with applications to quadratic programming and the minimum-norm problem. *Mathematical Programming*, 38:47–67, 1987.

[9] Michael P. Friedlander, Ives Macêdo, and Ting Kei Pong. Gauge optimization and duality. *SIAM Journal on Optimization*, 24(4):1999–2022, 2014.

[10] Alberto González-Sanz, Marcel Nutz, and Andrés Riveros Valdevenito. Monotonicity in quadratically regularized linear programs. *arXiv:2408.07871*, 2025.

[11] Hao Hao and Peter Zhang. Robust paths: Geometry and computation. *arXiv preprint arXiv:2508.20039*, 2025.

[12] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

[13] Peyman Mohajerin Esfahani and Daniel Kuhn. Data-driven distributionally robust optimization using the wasserstein metric: Performance guarantees and tractable reformulations. *Mathematical Programming*, 171(1):115–166, 2018.

[14] Karthik Natarajan, Dessislava Pachamanova, and Melvyn Sim. Constructing risk measures from uncertainty sets. *Operations Research*, 57(5):1129–1141, 2009.

[15] Melvyn Sim, Long Zhao, and Minglong Zhou. A new perspective on supervised learning via robust satisficing. *Available at SSRN 3981205*, 2021.

[16] Ningji Wei, Xian Yu, and Peter Zhang. Redefining coherent risk measures: From gauge optimization to regularization. *arXiv:2501.14989v2*, 2025.

[17] Eric Wong, Leslie Rice, and J. Zico Kolter. Fast is better than free: Revisiting adversarial training. In *International Conference on Learning Representations*, 2020.

[18] Weijun Xie. On distributionally robust chance constrained programs with wasserstein distance. *Mathematical Programming*, 186(1):115–155, 2021.

[19] Guanglin Xu and Grani A. Hanasusanto. Improved decision rule approximations for multistage robust optimization via copositive programming. *Operations Research*, 73(2):842–861, 2025.

Research Highlight
Spatial Branch-and-Bound for Nonconvex Separable Piecewise Linear Optimization

by Thomas Hübner[1], Akshay Gupte[2], Steffen Rebennack[3]
[1]Power Systems Laboratory, ETH Zürich, Switzerland
[2]School of Mathematics and Maxwell Institute for Mathematical Sciences, University of Edinburgh, United Kingdom
[3]Institute for Operations Research, Stochastic Optimization, Karlsruhe Institute of Technology (KIT)

We would like to thank the 2025 ICS Student Paper Award Committee members Ryan Cory-Wright, Austin Buchanan, Yongchun Li, and Young Woong Park for being selected as the runner-up for this award. We also extend our gratitude to the ICS Newsletter editor, Hamed Rahimian, for the invitation to share this research highlight with the ICS community. This summary is based on our work [6]

1 Introduction

We consider optimization problems with nonconvex separable piecewise-linear functions (PLFs) in the objective. A separable PLF is given by $F(x) = \sum_{i\in\mathcal{I}} f_i(x_i)$, where each $f_i : [l_i, u_i] \to \mathbb{R}$ is a univariate PLF (possibly discontinuous) over the interval $[l_i, u_i] \subset \mathbb{R}$. An example of such a function with five linear segments is shown in Figure 1. Our PLF optimization problem is stated as

$$z^* = \min \sum_{i=1}^{n} f_i(x_i) \quad \text{s.t.} \quad x \in S,\ x \in [l, u] \tag{1}$$

where S is a closed convex set and $[l, u] = \prod_{i=1}^{n}[l_i, u_i]$ is a box.

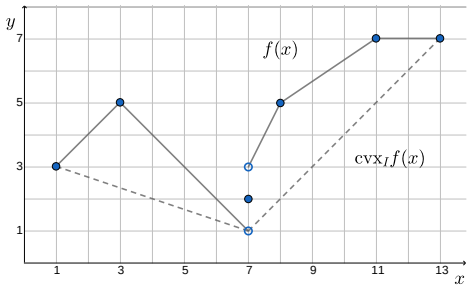


Figure 1: Univariate PLF (solid line) and its convex underestimator (dashed line) over the interval $[1, 13]$.

Optimization problems involving nonconvex separable PLFs in the objective function or constraints have long been studied in operations research, from early foundational work (e.g., [2]) to recent advances (e.g., [7, 8]). Furthermore, PLFs are commonly employed to linearize nonlinear terms and thereby, create a tractable (inner or outer) approximation to a nonconvex optimization problem. Best-fit PLFs can be determined computationally by solving (large-sized) MILPs [9], and there are also some adaptive methods to solve them efficiently. Small segments in the PLF give fine approximations of the problem, which may translate into sharp primal or dual bounds.

The standard approach for solving (1) is through special ordered sets (SOS) or mixed-integer linear programming (MILP) formulations of each f_i. In contrast, we develop a new spatial branch-and-bound (sBB) algorithm. Our approach is motivated by viewing PLF optimization through the lens of global optimization, due to nonconvexity of the PLF objective in (1). A basic sBB for optimizing separable functions was first proposed by [4], and since then these algorithms have matured tremendously. However, this global optimization approach has so far not been undertaken for PLF optimization, and state-of-the-art global solvers are unable to take PLFs directly as input without first being modeled using integer variables. Another drawback of existing methods is that they do not always scale well with the number of segments in the PLF. Our sBB algorithm has several desirable computational properties that allow it to solve problems with nonconvex separable PLFs considerably faster than MILP formulations when the number of segments of the PLFs is

sufficiently large. A method with good scalability properties is important because tight PLF approximations of nonlinear functions generally require a large number of segments.

The next section sketches the motivation and core ideas of our sBB algorithm. Subsequently, we present selected computational results and an error bound analysis of the number of breakpoints sufficient to approximate a nonconvex function with a continuous PLF achieving a desired error tolerance.

2 Spatial Branch-and-Bound

A lower semicontinuous (l.s.c.) PLF can be reformulated as a MILP using auxiliary continuous and binary variables [7, 10]. Thus, when each f_i is l.s.c., problem (1) can be reformulated as a mixed-integer convex program that can be solved by a general-purpose branch-and-cut solver. For strong computational performance, two properties are particularly desirable in MILP formulations of PLFs:

Small Size: A formulation should use few auxiliary variables and constraints.

Sharpness: A convex relaxation providing a strong lower bound can be constructed cheaply. The relaxation is called *sharp* if it achieves the best possible lower bound at a node, and is called *hereditarily sharp* if the sharpness persists at all nodes of the subtree after branching.

Sharpness and hereditary sharpness ensure that only a few nodes need to be processed because the gap closes faster. The small-size property ensures those nodes are processed quickly because the LP relaxation is small. Consequently, recent research on MILP formulations has focused on balancing small size and sharpness. While the logarithmic formulations of [11] significantly reduced size compared to classical approaches, they did so at the expense of hereditary sharpness. To address this, zigzag formulations were introduced by [7] to improve sharpness while preserving logarithmic size. Our motivation for introducing an sBB algorithm is to achieve both properties simultaneously. Rather than formulating each f_i as a MILP, our sBB algorithm computes cheap lower bounds at each node of the search tree through tractable convex relaxations.

Node Relaxations The obvious candidate for a relaxation is the perfect relaxation obtained by taking the convex envelope of the objective $\sum_i f_i$ over $S \cap [l, u]$, but this is hard in general even when S is a polyhedron. Instead, as is customary in standard sBB algorithms, at each node of the tree we consider the convex envelope of $F(x) := \sum_i f_i(x_i)$ over variable bounds corresponding to that node. Separability of the function and the domain of convexification implies that the envelope is the sum of envelopes in each dimension. Each univariate envelope is a PLF but may not be lower semicontinuous (l.s.c.) since we allow f_i's to be discontinuous. For computational tractability, we need the underestimators to be l.s.c. so that they have a polyhedral representation and yield a closed convex set as a relaxation. Hence, our underestimator is constructed by taking for each i, the convex envelope of the tightest l.s.c. function underestimating f_i over the bounds on x_i at that node.

Our underestimator, denoted by $\operatorname{vex}_{H'} F(x) := \sum_{i=1}^n \operatorname{vex}_{H'_i} f_i$ over any box $H' \subseteq [l, u]$, is a convex and continuous PLF and we give an explicit formula for it by characterising all its breakpoints [6, Proposition 4.1]. See Figure 1 for an illustration. Also, an efficient scanning-type algorithm [6, Proposition 4.2] is used to generate $\operatorname{vex}_{H'_i} f_i$ for each i by taking as input the function values and breakpoints of f_i over its domain H'_i. Thus, at any node of the sBB tree with corresponding box H' for variable bounds, we generate a lower bound by minimising $\sum_{i=1}^n \operatorname{vex}_{H'_i} f_i$ over $S \cap H'$; this problem can be reformulated in its epigraph form using an auxiliary continuous variable and multiple linear constraints (one for each segment of $\operatorname{vex}_{H'_i} f_i$ for each i). The size of our convex relaxations scales with the number of segments in the underestimator of each f_i. The epigraph formulation has sparsity due to two nonzeros in each constraint.

Fast node updates Besides generation of the convex underestimator $\operatorname{vex}_{H'} F$, another salient feature of our sBB is how this underestimator is updated from parent node to child node after branching. The efficiency of the sBB algorithm depends critically on this update being done efficiently. Since we branch on one variable at a time (i.e., simple disjunction) and the objective F is separable, it follows that at any node of the sBB tree, the underestimator is to be updated only in the dimension of the branched variable. Thus, if H^k is the box for variable bounds at a node created by branching on x_{i_k} from parent node box H^{k-1}, then the convex relaxation we solve at this node is

$$v(H^k) := \min \sum_{i \neq i_k} \operatorname{vex}_{H_i^{k-1}} f_i(x_i) + \operatorname{vex}_{H_{i_k}^k} f_{i_k}(x_{i_k}) \quad \text{s.t.} \quad x \in S \cap H^k \tag{2}$$

We can apply our scanning method [6, Algorithm 4.1] to generate the underestimator of f_{i_k} from scratch at this node, but this is not efficient and will not make the sBB converge faster. Instead, for a quick and fast update, we exploit computational geometry of univariate PLFs to show [6, Proposition 4.3] that the underestimator of a univariate PLF over a subinterval does not change between the leftmost and rightmost breakpoints in the subinterval, which can lead to substantial savings in computation if the subinterval at child node is large w.r.t. interval at parent node. In particular, if H^k is the $\leq$-branch from H^{k-1} by branching $x_{i_k} \leq \bar{x}_{i_k}$, and $b_{i_k}^{k,up}$ is the largest breakpoint of $\text{vex}_{H_{i_k}^{k-1}} f_{i_k}$ that is no more than $\bar{x}_{i_k}$, then $\text{vex}_{H_{i_k}^{k}} f_{i_k}$ differs from $\text{vex}_{H_{i_k}^{k-1}} f_{i_k}$ only over the interval $[b_{i_k}^{k,up}, \bar{x}_{i_k}]$. Hence, the most efficient way to generate the underestimator at this node is to use the breakpoints of $\text{vex}_{H_{i_k}^{k-1}} f_{i_k}$ stored in memory and compute new breakpoints over $[b_{i_k}^{k,up}, \bar{x}_{i_k}]$ using our scanning method [6, Algorithm 4.1]. The $\geq$-branch is solved analogously.

Convergence Our sBB performs branching on continuous variables (one at a time) to achieve convergence. Spatial branching offers a higher degree of flexibility in branching decisions compared with integer or SOS2 branching. Both integer and spatial branching choose a branching variable; but the latter can branch at any point within the interval of the variable bounds, whereas integer branching in MILP and SOS2 models can be mapped to specific points in each interval. Therefore, spatial branching can mimic integer and SOS2 branching, but the converse is not true.

At any node k in the sBB tree, if the node relaxation (2) has returned an optimal solution $\bar{x}$, then we choose a branching variable x_{i_k} through some criterion and split the interval $[l_{i_k}^k, u_{i_k}^k]$ of x_{i_k} at the current node into two child nodes with intervals $[l_{i_k}^k, \bar{x}_{i_k}]$ and $[\bar{x}_{i_k}, u_{i_k}^k]$. We consider three rules for selecting branching variable i_k based on (1) largest error between f_i and its underestimator $\text{vex}_{H_i^k} f_i$, either across the entire interval of variable bounds or only at breakpoints, or (2) widest interval (longest edge) of variable bounds. We show finite or asymptotic convergence [6, Propositions 5.1 and 5.2] for largest error rules, whereas the longest-edge rule is known [12] to converge asymptotically under a strongly consistent assumption.

We also update upper bounds using value of objective F at node solutions, and prune nodes when their relaxation value is ε-close to the incumbent upper bound. Our complete sBB algorithm is in [6, Algorithm 5.1]. The next result summarises our main theoretical properties.

Theorem 8. *The sBB algorithm, where node relaxations are computed using the tightest convex continuous underestimator in each dimension and updated between nodes by exploiting the subinterval property, converges finitely using the breakpoint branching rule when each f_i is l.s.c., and converges asymptotically using the largest-error branching rule when each f_i is continuous.*

3 Computational Results

We compare a rudimentary Python implementation of our sBB algorithm against five standard MILP approaches. These are Gurobi's built-in PLF solver (GRB) and four state-of-the-art logarithmic-sized MILP formulations — *Logarithmic* (Log) and *Disaggregated Logarithmic* (DLog) by [10] and *Binary Zig-Zag* (ZZB) and *General Integer Zig-Zag* (ZZI) by [7], which are available in the *PiecewiseLinearOpt* package by [7]. The MILP models are generated in Julia and solved using Gurobi.

We mention here our numerical results on network flow problems with continuous concave PLFs,

$$\min \sum_{i=1}^{n}\sum_{j=1}^{n} f_{ij}(x_{ij}) \quad \text{s.t.} \quad \sum_{j=1}^{n} x_{ij} - \sum_{j=1}^{n} x_{ji} = d_i \quad i = 1,\ldots,n, \quad l_{ij} \leq x_{ij} \leq u_{ij} \quad i,j = 1,\ldots,n.$$

More computations with knapsack problems and l.s.c. PLFs are in our paper [6]. Instances are generated similarly to prior studies by [7, 10]. Our experiments focus on problems with $n = 10$ nodes. Let K denote the number of linear segments in each PLF f_{ij}. For each value of K, we generate and solve 50 random instances. Table 1 summarizes solve-time statistics, including median (med.), arithmetic mean (avg.), standard deviation (std.), the number of instances not solved within the time limit (fail), and the number of instances for which each method was the fastest (win).

Thus, the sBB algorithm scales exceptionally well with the number of PLF segments K, whereas the MILP formulations scale much less favourably. For PLFs with around $K = 500$ segments, the computational advantages that MILP models gain from being solved within the commercial solver Gurobi are offset by the

Table 1: Solve times [sec.] for network flow problems with continuous concave PLFs. The methods are sorted from fastest to slowest in terms of median runtime. Our sBB algorithm is marked in bold.

Method	Med.	Avg.	Std.	Win	Fail
a) 10 segments					
ZZI	**0.26**	0.28	0.12	20	0
Log	**0.26**	0.34	0.50	12	0
DLog	**0.33**	0.32	0.14	9	0
ZZB	**0.39**	0.41	0.22	4	0
GRB	**0.40**	0.36	0.14	5	0
sBB	**4.55**	8.96	14.65	0	0
b) 100 segments					
ZZI	**1.82**	2.11	1.06	24	0
ZZB	**1.95**	2.30	1.15	15	0
Log	**2.36**	2.68	1.19	5	0
DLog	**3.90**	4.67	2.20	0	0
sBB	**4.11**	6.17	7.08	6	0
GRB	**9.38**	9.65	3.95	0	0
c) 500 segments					
sBB	**8.2**	11.1	11.2	39	0
Log	**15.1**	17.2	8.0	6	0
ZZI	**15.5**	18.0	9.0	4	0
ZZB	**15.8**	19.4	11.9	1	0
DLog	**23.6**	27.6	14.6	0	0
GRB	**90.0**	114.6	94.8	0	0

Method	Med.	Avg.	Std.	Win	Fail
d) 1,000 segments					
sBB	**5.8**	10.9	14.6	50	0
Log	**45.7**	49.5	18.2	0	0
DLog	**46.5**	57.4	32.4	0	0
ZZI	**48.4**	48.9	21.1	0	0
ZZB	**61.6**	61.2	23.9	0	0
GRB	**270.6**	363.4	241.9	0	0
e) 5,000 segments					
sBB	**7.2**	11.0	11.0	50	0
Log	**330**	331	132	0	0
ZZI	**333**	329	152	0	0
DLog	**379**	405	200	0	0
ZZB	**515**	518	198	0	0
GRB	**1,800**	1,800	0	0	50
f) 10,000 segments					
sBB	**8**	12	15	50	0
Log	**729**	763	294	0	0
DLog	**940**	876	374	0	1
ZZI	**976**	924	299	0	1
ZZB	**1,419**	1,368	410	0	9
GRB	**1,800**	1,800	0	0	50

strengths of our sBB approach – namely, its compact LP relaxations and consistently sharp lower bounds – even though our implementation is only a simple Python prototype. Extensive computational results in our paper further demonstrate clear computational advantages of the slim and sharp bounds in our sBB when the number of segments increases.

4 Error Analysis

A key question when building PLF approximations is to determine how many pieces each PLF should have if the approximation error, defined as the largest distance between the function value and the approximate value, is to be no more than some given error bound. Such error-bounding analysis has been done for bilinear terms [1, 3]. For a thrice-continuously differentiable univariate function, an asymptotic answer is known [5] in the sense that the number of breakpoints in a PLF achieving an error of ε is roughly $c/\sqrt{\varepsilon}$ as $\varepsilon \to 0$, where the constant c depends on some second-order derivative of the function. However, there is no result estimating the number of breakpoints for any $\varepsilon > 0$ without requiring differentiability. We present such an error-bounding analysis for a large class of continuous functions.

Proposition 3 ([6, Proposition 7.2])**.** *For any (α, β)-Hölder continuous univariate function over $[0, 1]$, to obtain a continuous PLF approximation with all but two breakpoints spaced at least $\delta \in (0, 1)$ apart and with additive error at most $\varepsilon > 0$, it suffices to have $(1 - \sqrt[\alpha]{\tau})/\delta$ many segments in the PLF, where $\tau := \delta\,\varepsilon/2\beta$.*

We implicitly assume $\alpha \in (0, 1]$ because a Hölder continuous function is constant for $\alpha > 1$. The actual lower bound on the number of breakpoints is a bit smaller and can be found in our paper. Note that with the above bound we may have one segment of the discretisation be of length less than δ because $\tau < \delta^\alpha$.

Since we are working with additive errors, the above result can be glued together across different dimensions when optimising a separable function, to deduce an error bound on a good PLF approximation for the problem. Define $\tau_i := \delta\,\varepsilon/2\beta_i$, for dimension $i = 1, \ldots, n$.

Corollary 1 ([6, Proposition 7.1])**.** *When minimising a sum of n univariate Hölder continuous functions over a convex set in $[0, 1]^n$, given any $\delta \in (0, 1)$ and $\varepsilon > 0$, the continuous PLF arising from creating $\left(1 - \sqrt[\alpha_i]{\tau_i/n}\right)/\delta$*

segments, spaced at least δ apart, in each dimension i yields an approximate value whose difference to the global minimum is no more than ε.

5 Conclusions

We have proposed a new approach to separable PLFs from the perspective of global and nonlinear continuous optimization. The developed spatial branch-and-bound algorithm has small, sparse, and sharp relaxations throughout the search tree. Computational experiments have shown that even a rudimentary sBB implementation in Python can outperform state-of-the-art logarithmic models if the number of segments is sufficiently high. We have also provided a rigorous justification for the need to have many segments when constructing low error PLF approximations of a nonconvex continuous function. Nonetheless, we advocate a problem-specific approach when selecting a solution method for separable PLFs. If many segments are involved, the sBB could be the method of choice because of its slim and sparse LP relaxations, whereas with few segments, MILP models, such as the classical incremental model or advanced logarithmic models, might be faster while also allowing strong relaxations through cutting planes.

Our rudimentary implementation can benefit from several enhancements and sophistications that would accelerate its performance. Further research can focus on extensions to nonseparable cases, cutting planes, specialized branching rules, integration in a full branch-and-cut solver, or convergent sBB algorithms for non-l.s.c. functions.

References

[1] Bärmann A, Burlacu R, Hager L, Kleinert T (2023) On piecewise linear approximations of bilinear terms: structural comparison of univariate and bivariate mixed-integer programming formulations. *Journal of Global Optimization* 85(4):789–819.

[2] Beale EM, Forrest JJ (1976) Global optimization using special ordered sets. *Mathematical Programming* 10(1):52–69.

[3] Dey SS, Gupte A (2015) Analysis of MILP techniques for the pooling problem. *Operations Research* 63(2):412–427.

[4] Falk JE, Soland RM (1969) An algorithm for separable nonconvex programming problems. *Management Science* 15(9):550–569.

[5] Frenzen CL, Sasao T, Butler JT (2010) On the number of segments needed in a piecewise linear approximation. *Journal of Computational and Applied Mathematics* 234(2):437–446.

[6] Hübner T, Gupte A, Rebennack S (2026) Spatial branch-and-bound for nonconvex separable piecewise linear optimization. *INFORMS Journal on Computing* 38(2):645–675.

[7] Huchette J, Vielma JP (2023) Nonconvex piecewise linear functions: Advanced formulations and simple modeling tools. *Operations Research* 71(5):1835–1856.

[8] Lyu B, Hicks IV, Huchette J (2026) Building formulations for piecewise linear relaxations of nonlinear functions. *Operations Research* 74(1):484–499.

[9] Rebennack S, Krasko V (2020) Piecewise linear function fitting via mixed-integer linear programming. *INFORMS Journal on Computing* 32(2):507–530.

[10] Vielma JP, Ahmed S, Nemhauser G (2010) Mixed-integer models for nonseparable piecewise-linear optimization: Unifying framework and extensions. *Operations Research* 58(2):303–315.

[11] Vielma JP, Nemhauser GL (2011) Modeling disjunctive constraints with a logarithmic number of binary variables and constraints. *Mathematical Programming* 128(1):49–72.

[12] Wechsung A, Barton PI (2014) Global optimization of bounded factorable functions with discontinuities. *Journal of Global Optimization* 58(1):1–30.

Research Highlight
The Surprising Performance of Random Partial Benders Decomposition

by JEAN PAUPHILET[1], YUPENG WU[1]
[1]LONDON BUSINESS SCHOOL

We would like to thank the committee of the 2025 INFORMS Computing Society Student Paper Award for highlighting our work. This summary is based on Pauphilet and Wu [9].

1 Introduction

Mixed-integer optimization [8] is a powerful mathematical framework for problems involving discrete decisions. Despite orders of magnitude speed-ups in both hardware and algorithms over the last decades [2], integer variables remain a central difficulty in large-scale optimization.

For large Mixed-Integer Linear Optimization (MILO) problems, Benders [1] proposed a decomposition scheme, known as Benders Decomposition (BD), that decomposes the problem into a pure-integer master problem and a continuous separation problem. On the modeling side, BD projects out the continuous variables and obtains a pure-integer convex optimization problem. On the algorithmic side, BD solves this formulation by iteratively building a piecewise linear outer approximation of the convex cost function. BD has been successful in production planning, facility location, and network design problems, and remains a central technique for large-scale optimization.

Formally, consider a generic MILO problem of the form

$$\min_{\boldsymbol{y}\in\mathcal{P}} \min_{\boldsymbol{x}_r, r\in\mathcal{R}} \quad \boldsymbol{d}^\top \boldsymbol{y} + \sum_{r\in\mathcal{R}} \boldsymbol{c}_r^\top \boldsymbol{x}_r \qquad \text{s.t.} \quad \boldsymbol{A}_r\boldsymbol{x}_r + \boldsymbol{B}_r\boldsymbol{y} \geq \boldsymbol{b}_r, \quad \forall r\in\mathcal{R}, \boldsymbol{x}_r \in \mathbb{R}^{N_r}_+. \tag{P}$$

where $\mathcal{P} := \{\boldsymbol{y}\in\{0,1\}^m \mid \boldsymbol{G}\boldsymbol{y}\geq\boldsymbol{g}\}$. Problem (P) is best understood as a two-stage problem: the decision maker first makes a binary design decision $\boldsymbol{y}$, followed by second-stage continuous decisions $\boldsymbol{x}_r$. For a fixed $\boldsymbol{y}$, the continuous variables decompose into $|\mathcal{R}|$ independent subproblems. This structure appears in a range of problems including stochastic Multi-commodity Capacitated Network Design (MCND), Uncapacitated Facility Location (UFL), and Maximum Coverage Facility Location (MCFL).

BD reformulates Problem (P) as

$$\min_{\boldsymbol{y}\in\mathcal{P}} \quad \boldsymbol{d}^\top\boldsymbol{y} + \sum_{r\in\mathcal{R}} \phi_r(\boldsymbol{y}), \qquad \text{with} \quad \phi_r(\boldsymbol{y}) := \min_{\boldsymbol{x}_r\geq 0} \left\{\boldsymbol{c}_r^\top\boldsymbol{x}_r \mid \boldsymbol{A}_r\boldsymbol{x}_r + \boldsymbol{B}_r\boldsymbol{y}\geq\boldsymbol{b}_r\right\}. \tag{1}$$

Each function ϕ_r is convex as the point-wise maximum of affine functions obtained by strong duality. BD replaces $\sum_r \phi_r(\boldsymbol{y})$ by a piecewise linear lower approximation and refines it by adding Benders cuts.

In this work, we propose a simple, generic, yet highly effective method to accelerate BD, which we call Random Partial Benders Decomposition (RPBD). The key modeling decision in RPBD is to keep a randomly selected subset of the continuous variables in the master problem. This makes the master problem larger than in BD but more informative. Our main observation is that retaining even a small random subset often provides enough information to recover much of the strength of the original MILO formulation.

2 Method

The separability in Problem (P) allows us to keep a subset of the second-stage continuous variables in the master problem and partially minimize with respect to the remaining ones. Given $\mathcal{R}^{\text{mp}} \subseteq \mathcal{R}$, RPBD solves

$$\min_{\boldsymbol{y}\in\mathcal{P},\, \boldsymbol{x}_r\geq 0,\, r\in\mathcal{R}^{\text{mp}}} \left\{ \boldsymbol{d}^\top\boldsymbol{y} + \sum_{r\in\mathcal{R}^{\text{mp}}} \boldsymbol{c}_r^\top\boldsymbol{x}_r + \sum_{r\in\mathcal{R}\setminus\mathcal{R}^{\text{mp}}} \phi_r(\boldsymbol{y}) \;\middle|\; \boldsymbol{A}_r\boldsymbol{x}_r + \boldsymbol{B}_r\boldsymbol{y}\geq\boldsymbol{b}_r,\ r\in\mathcal{R}^{\text{mp}} \right\}. \tag{2}$$

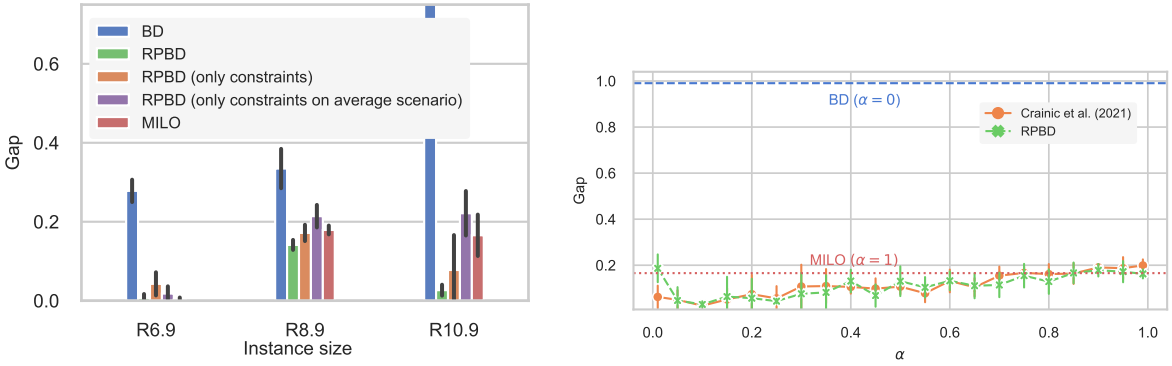
We then apply the same outer-approximation procedure as BD, but only to the functions ϕ_r with $r \in \mathcal{R} \setminus \mathcal{R}^{\text{mp}}$. Problem (2) reduces to BD when $\mathcal{R}^{\text{mp}} = \emptyset$ and to the original MILO formulation when $\mathcal{R}^{\text{mp}} = \mathcal{R}$.

The intuition is a tightness–tractability trade-off. The classical BD master problem is small but hides the second-stage information; the original MILO formulation is exact but can be too large. RPBD interpolates between these extremes. The retained variables strengthen the relaxation and make solver heuristics, branching decisions, and feasibility searches more informative.

Our selection rule is problem-agnostic: $\mathcal{R}^{\text{mp}}$ is constructed by sampling a fraction α of the indices in $\mathcal{R}$ without replacement. The only hyper-parameter is α. For a given instance, we calibrate it at the root node level by gradually increasing α along a predefined grid and selecting the largest value whose root node relaxation can be solved within a small fraction of the overall time limit. For MCND, we also compare against the feasibility-based partial BD rule of Crainic et al. [6].

3 Numerical Results

We evaluate RPBD on MCND, UFL, and MCFL. For MCND, we use the large and hard $\mathbf{R}^\star$ instances from Crainic et al. [5]; the largest size, R10.9, has 20 nodes, 120 edges, and 40 commodities. For UFL, we use the $\mathbf{M}^\star$ instances from Kratica et al. [7]. For MCFL, we generate instances following Cordeau et al. [4], ReVelle et al. [10]. We compare RPBD with solving the MILO formulation and with a standard single-cut BD. All methods use the same initial feasible solution, valid inequalities, and default solver settings.



(a) Gap at termination on MCND.

(b) Comparison with Crainic et al. [6] on R10.9.

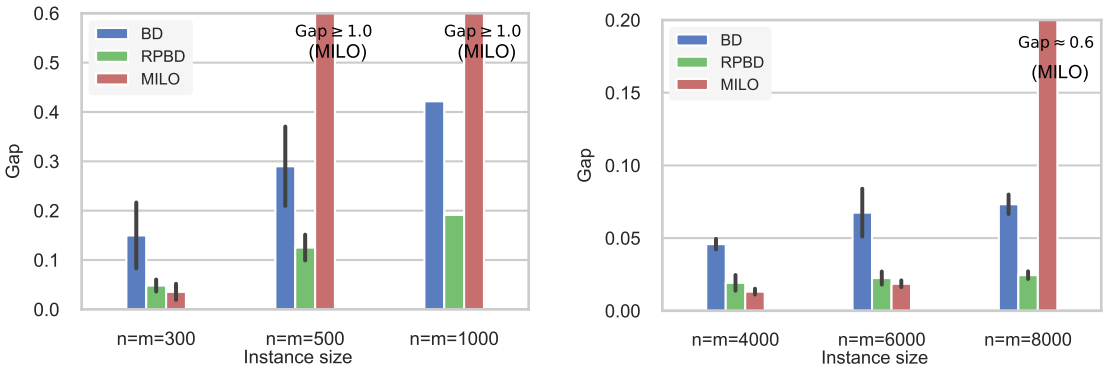
Figure 1: Comparative performance on MCND. Panel (a) reports average optimality gaps at termination (3-minute time limit) for BD, RPBD, MILO, as well as two RPBD variants. Panel (b) compares RPBD with the feasibility-based rule of Crainic et al. [6]. Error bars represent standard deviations.

Figure 1(a) reports the average optimality gap at termination on MCND instances. Comparing MILO, BD, and RPBD: BD performs worse than MILO because MCND does not satisfy relatively complete recourse and BD spends many iterations generating feasibility cuts. RPBD substantially improves upon BD. On the largest R10.9 instances, RPBD is the clear winner, reaching an average gap of 0.03 versus 0.16 for MILO.

Figure 1(b) compares RPBD with the feasibility-based rule of Crainic et al. [6] on R10.9. RPBD performs almost identically across values of α, with comparable average gaps and overlapping error bars, even though it uses no problem-specific information. Note that we excluded from our time limit the time needed to compute the set $\mathcal{R}^{\text{mp}}$, which can be substantial for the feasibility-based rule of Crainic et al. [6].

On problems without relatively complete recourse, RPBD can help for two reasons: it enforces some second-stage feasibility constraints and it better approximates the second-stage cost. To isolate the relative contribution of each mechanism, we also compare the performance of BD/RPBD in Figure 1(a) with two additional benchmarks: one where we introduce the second-stage variables $\boldsymbol{x}_r, r \in \mathcal{R}^{\text{mp}}$ and second-stage constraints, but we keep the epigraph variable η_r and use Benders cuts to approximate the second-stage cost ('RPBD (only constraints)'); and one where we introduce a second-stage variable $\bar{\boldsymbol{x}}$ and second-stage feasibility constraints associated with the average demand scenario ('RPBD (only constraints on average scenario)'). In our numerical results, RPBD still outperforms the two variants, thereby highlighting that the value of RPBD is not limited to enforcing feasibility.

We also evaluate the performance of RPBD on problems that satisfy relatively complete recourse, namely



(a) Gap at termination on UFL.

(b) Gap at termination on MCFL.

Figure 2: Comparative performance on UFL and MCFL. Both panels report average optimality gaps at termination for BD, RPBD, and MILO. Error bars represent standard deviations.

UFL and MCFL. RPBD remains effective; see Figure 2(a)–(b). On larger UFL instances, it achieves average gaps of 0.13 and 0.19 for $n = m = 500$ (n: number of customers; m: number of locations) and $n = m = 1000$, compared with 0.29 and 0.42 for the second-best method. On MCFL with $n = m = 8000$, RPBD reaches a gap three times lower than the best benchmark.

4 Theoretical Analysis

We provide a theoretical explanation for the strength of the lower bounds obtained by RPBD. Consider one iteration of BD, where each convex term ϕ_r is approximated by a piecewise linear lower approximation $\hat{\phi}_r$. Let $\mathcal{P}_c$ denote the Boolean relaxation of the current branch-and-bound node. The node relaxations obtained by the MILO and BD formulations are

$$v^\star := \min_{\boldsymbol{y}\in\mathcal{P}_c} \sum_{r\in\mathcal{R}} \phi_r(\boldsymbol{y}), \qquad \hat{v}^\star := \min_{\boldsymbol{y}\in\mathcal{P}_c} \sum_{r\in\mathcal{R}} \hat{\phi}_r(\boldsymbol{y}),$$

where we omit the first-stage cost because it can be subsumed in the functions ϕ_r and $\hat{\phi}_r$. Let $\boldsymbol{u}^\alpha \in \{0,1\}^{\mathcal{R}}$ encode the retained indices, with $u^\alpha_r = 1$ iff $r \in \mathcal{R}^{\text{mp}}$. The RPBD lower bound is

$$\hat{v}^\star(\boldsymbol{u}^\alpha) := \min_{\boldsymbol{y}\in\mathcal{P}_c} \sum_{r\in\mathcal{R}} (1-u^\alpha_r)\hat{\phi}_r(\boldsymbol{y}) + u^\alpha_r \phi_r(\boldsymbol{y}),$$

and $\hat{v}^\star \le \hat{v}^\star(\boldsymbol{u}^\alpha) \le v^\star$. To quantify the improvement, we introduce the deterministic expected counterpart

$$\bar{v}(\alpha) := \min_{\boldsymbol{y}\in\mathcal{P}_c} \sum_{r\in\mathcal{R}} (1-\alpha)\hat{\phi}_r(\boldsymbol{y}) + \alpha\phi_r(\boldsymbol{y}).$$

First, we show that the expected counterpart is a good proxy for the RPBD bound, especially for small α.

Proposition 4. *Denote $\Delta_{\max}$ an upper bound on the approximation error at the RPBD solutions, i.e., $\Delta_{\max} = \max_r \max_{\boldsymbol{u}}(\phi_r - \hat{\phi}_r)(\boldsymbol{y}^\star(\boldsymbol{u}))$, where $\boldsymbol{y}^\star(\boldsymbol{u})$ denotes an optimal solution of $\hat{v}^\star(\boldsymbol{u})$. For any $\alpha \in [0,1]$, we have $\mathbb{E}[\hat{v}^\star(\boldsymbol{u}^\alpha)] \le \bar{v}(\alpha)$ and $\bar{v}(\alpha) - \alpha(1-\alpha)\Delta_{\max}|\mathcal{R}| \le \hat{v}^\star(\boldsymbol{u}^\alpha)$.*

Then, we relate the value of the expected counterpart to the naive lower bound $(1-\alpha)\hat{v}^\star + \alpha v^\star$.

Proposition 5. *For any $\alpha \in [0,1]$, we have $(1-\alpha)\hat{v}^\star + \alpha v^\star + \rho(\alpha) \le \bar{v}(\alpha) \le v^\star$, where $\rho : [0,1] \to \mathbb{R}$ satisfies: (i) $\rho(\alpha)$ is piecewise linear and concave; (ii) $\rho(\alpha) = 0$ if $\alpha = 0$ or 1; and (iii) $\rho'(0) = \sum_{r\in\mathcal{R}} \phi_r(\hat{\boldsymbol{y}}) - v^\star \ge 0$, the sub-optimality of the relaxed solution of $\hat{v}^\star$.*

Proposition 5 and Figure 3 state that the expected counterpart can be much tighter than a simple interpolation between the BD and MILO relaxations. In particular, for small α, RPBD can significantly strengthen the lower bound whenever the solution found by the BD relaxation is sub-optimal for the MILO relaxation.

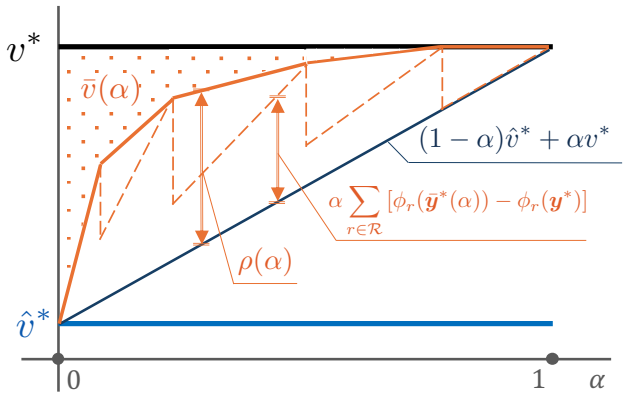


Figure 3: Illustration of Proposition 5.

Conclusion

RPBD accelerates BD by randomly retaining a subset of continuous variables in the master problem. The method is simple to implement, problem-agnostic, and effective across problems with and without relatively complete recourse. Numerically, it substantially reduces optimality gaps on large MCND, UFL, and MCFL instances and matches problem-specific partial BD rules on the settings for which those rules were designed. Theoretically, RPBD strengthens the relaxation value by replacing part of the Benders outer approximation with exact second-stage structure.

A promising direction is to use RPBD for two-stage stochastic problems with mixed-integer recourse. In this setting, BD applies after relaxing the integrality of the second-stage variables and only provides a relaxation to the original problem. While RPBD remains a relaxation, preliminary results suggest that it can produce much tighter bounds and better feasible solutions, hence increasing the scope of instances that can be solved to reasonable accuracy without the need for stronger integer-recourse cuts [as developed in, e.g., 3].

References

[1] Benders JF (1962) Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik* 4(1):238–252.

[2] Bixby RE (2012) A brief history of linear and mixed-integer programming computation. *Documenta Mathematica* 2012:107–121.

[3] Chen R, Luedtke J (2022) On generating lagrangian cuts for two-stage stochastic integer programs. *INFORMS Journal on Computing* 34(4):2332–2349.

[4] Cordeau JF, Furini F, Ljubić I (2019) Benders decomposition for very large scale partial set covering and maximal covering location problems. *European Journal of Operational Research* 275(3):882–896.

[5] Crainic TG, Fu X, Gendreau M, Rei W, Wallace SW (2011) Progressive hedging-based metaheuristics for stochastic network design. *Networks* 58(2):114–124.

[6] Crainic TG, Hewitt M, Maggioni F, Rei W (2021) Partial Benders decomposition: general methodology and application to stochastic network design. *Transportation Science* 55(2):414–435.

[7] Kratica J, Tošic D, Filipović V, Ljubić I (2001) Solving the simple plant location problem by genetic algorithm. *RAIRO-Operations Research* 35(1):127–142.

[8] Nemhauser GL, Wolsey LA (1999) *Integer and Combinatorial Optimization* (John Wiley & Sons).

[9] Pauphilet J, Wu Y (2025) The surprising performance of random partial benders decomposition. *Optimization Online* 32181.

[10] ReVelle C, Scholssberg M, Williams J (2008) Solving the maximal covering location problem with heuristic concentration. *Computers & Operations Research* 35(2):427–435.