A Dual Perspective on Decision-Focused Learning

Paula Rodriguez-Diaz
Harvard University

Kirk Bansak
University of California, Berkeley

Elisabeth Paulson
Harvard University

Abstract

Predict-then-optimize (PtO) pipelines use predictions as inputs to a downstream optimizer that produces decisions. Decision-Focused Learning (DFL) trains the predictor inside this pipeline to improve the quality of those decisions, not just prediction accuracy. Most DFL approaches do so by differentiating through the optimizer or by designing tight task-specific surrogates—both of which demand frequent solver calls and drive up training cost. We introduce a dual-guided DFL method that preserves decision alignment while sharply reducing solver-in-the-loop cost. The key idea is to solve the downstream problem only periodically to refresh dual variables and, between refreshes, train on dual-adjusted targets using simple surrogate losses. As refreshes become less frequent, the training loop's cost approaches standard supervised learning while maintaining high-quality decisions.

1 Introduction

In many real-world decision-making settings, decisions are made under uncertainty: forecasts feed a downstream optimizer that selects actions. PtO formalizes this workflow, with predictions fed into a downstream solver that returns decisions. The prevailing approach is a two-stage pipeline: (i) train a predictor to minimize a statistical loss (e.g., mean-squared error or negative log-likelihood), then (ii) plug those predictions into an optimization routine to pick actions. This setup optimizes for predictive accuracy, not decision quality, and the two objectives can diverge: errors are weighted uniformly rather than by their impact on feasibility and cost; small errors near decision boundaries can flip the chosen action and incur large regret; and problem-specific asymmetries and constraints are ignored.

To address this mismatch, DFL methods train the predictor inside this pipeline to prioritize downstream decision quality over prediction accuracy alone [Mandi et al., 2024]. DFL is typically achieved through two routes. One embeds a *differentiable optimization layer* so the model is trained end-to-end through the solver (e.g., OptNet; CVXPYLayers), which requires a forward solve and backpropagation through the optimization at each update [Wilder et al., 2019, Amos and Kolter, 2017]. The other replaces direct regret minimization with *surrogate objectives* (e.g., SPO+) that approximate decision loss but still invoke one or more solves per batch/iteration (often on loss-augmented or smoothed problems) [Elmachtoub and Grigas, 2022]. In both cases, optimization is invoked repeatedly throughout training, validation, and tuning, which creates substantial computational and memory overhead [Mandi et al., 2020, Shah et al., 2022]. These pressures are especially acute for combinatorial tasks—assignments, matchings, knapsack, packing, routing—where exact solves are expensive and even relaxed or approximate oracles remain costly at training cadence. Consequently, runtime and scalability are central obstacles for deploying DFL on realistic data and problem sizes [Wang et al., 2023].

In this work we address the tension between DFL and efficiency. Our key idea is that dual information, obtained from a single solve, can guide many subsequent gradient steps without re-invoking the solver. Concretely, we refresh dual variables only periodically, and between refreshes we train on dual-adjusted targets using simple surrogate losses. This decouples the frequency of optimization from the update loop, sharply reducing overhead while preserving decision awareness. We show that dual-guided learning dramatically improves scalability and runtime efficiency, especially in combinatorial settings, while maintaining competitive task performance. By addressing the core

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: MLxOR: Mathematical Foundations and Operational Integration of Machine Learning for Uncertainty-Aware Decision-Making.

bottleneck of repeated solves, our framework takes a step toward making decision-focused methods practical for larger-scale and time-sensitive applications.

2 Dual-Guided Loss

We consider binary linear programs with natural "pick-one" groups, formulated as:

$$\max_{x\in\{0,1\}^n} \quad y^\top x \quad \text{s.t.} \quad Ax \le b \quad \text{and} \quad \sum_{i\in g} x_i = 1 \quad \forall g \in \mathcal{G}. \tag{1}$$

Here, $A \in \mathbb{R}^{m\times n}$, $\mathbf{b} \in \mathbb{R}^m$, and $\mathcal{G}$ partitions $\{1,\dots,n\}$ into mutually-exclusive groups. Many classical combinatorial problems can be expressed in this form, including assignment, shortest path, and knapsack (see Appendix). Notably, five of the seven benchmark tasks in the recent DFL survey by Mandi et al. [2024] can be expressed in this form.

We look at this problem structure in the context of PtO, where the cost vector $\mathbf{y}$ is not directly observable and must be estimated from contextual features using a predictive model. The estimated $\mathbf{y}$ is then used to solve the decision problem.

To construct our decision-focused loss, let $\lambda \ge 0$ be the dual variables for the global constraint $Ax \le b$ in the continuous relaxation of equation 1. The corresponding Lagrangian is

$$\mathcal{L}(x,\lambda) = y^\top x + \lambda^\top(b - Ax) = \left(y - A^\top\lambda\right)^\top x + \lambda^\top b.$$

Because the problem includes "pick-one" constraints, a natural candidate solution to

$$\max_{x\in\{0,1\}^n} \quad \mathcal{L}(x,\lambda) \quad \text{s.t.} \sum_{i\in g} x_i = 1 \quad \forall g \in \mathcal{G} \tag{2}$$

is given by $x_j = 1$ if and only if $j = \arg\max_i y_i - (A^\top\lambda)_i$ for $i \in g,\ g \in \mathcal{G}$ (ties broken arbitrarily). This motivates defining the *dual-adjusted scores*, $h_i(\mathbf{y},\lambda) \triangleq y_i - (A^\top\lambda)_i$. To make this selection differentiable, we apply a temperature-controlled softmax within each group g:

$$\tilde{z}_{g,\tau}(\hat{\mathbf{y}},\lambda) \triangleq \text{softmax}((h_i(\hat{\mathbf{y}},\lambda))_{i\in g},\tau) \triangleq \left(\frac{\exp(h_j(\hat{\mathbf{y}},\lambda)/\tau)}{\sum_{i\in g}\exp(h_i(\hat{\mathbf{y}},\lambda)/\tau)}\right)_{j\in g},$$

which converges to $\arg\max$ as $\tau \to 0$. For comparison, let $z^*_g(\hat{\mathbf{y}},\lambda) = \mathbf{e}_{\arg\max(h_i(\hat{\mathbf{y}},\lambda))_{i\in g}}$ denote the discrete $\arg\max$ assignment for group g. Finally, we define the *Dual-Guided Loss* (DGL) as

$$\mathcal{L}_\tau \triangleq -\frac{1}{|\mathcal{G}|}\sum_{g\in\mathcal{G}} \mathbf{y}^\top \tilde{z}_{g,\tau}(\hat{\mathbf{y}},\lambda). \tag{3}$$

This loss drives $\hat{\mathbf{y}}$ to produce decisions $\tilde{z}_\tau(\hat{\mathbf{y}},\lambda)$ that score well under the dual-adjusted reward $\mathbf{y} - \lambda$, approximating the true-cost optimum while implicitly encoding constraints via λ. It is fully differentiable and avoids repeated solves and solver backpropagation.

2.1 Learning with DGL

Let $M_\theta : \mathcal{X} \to \mathbb{R}^d$ denote a standard prediction model parameterized by θ (e.g., a neural network) that maps features $\mathbf{x} \in \mathcal{X}$ to predicted costs $\hat{\mathbf{y}} = M_\theta(\mathbf{x})$. We train M_θ by minimizing the DGL with gradient descent: for each feature vector $\mathbf{x}$, the model predicts $\hat{\mathbf{y}} = M_\theta(\mathbf{x})$, and we form soft surrogate decisions $\tilde{z}_\tau(\hat{\mathbf{y}},\lambda)$, enabling full backpropagation through this differentiable surrogate. A key challenge is handling the dual variables λ, computed from reference costs $\mathbf{y}$ or $\hat{\mathbf{y}}$. As training progresses, $\hat{\mathbf{y}}$ may move toward regions that improve decision quality yet deviate from the true costs $\mathbf{y}$. When this happens, duals computed once from $\mathbf{y}$ may no longer reflect the relevant constraint interactions for $\hat{\mathbf{y}}$. To address this, we explore strategies that periodically refresh duals during training:

- **No-update:** Duals are computed once at initialization by solving the continuous relaxation of equation 1 with ground-truth costs $\mathbf{y}$, and then kept fixed for the duration of training.
- **Fixed-frequency:** Duals are recomputed for all training instances every U epochs using the current predictions $\hat{\mathbf{y}} = M_\theta(\mathbf{x})$. This balances computational cost with adaptability.
- **Auto-update:** Duals are updated only for instances where surrogate decision (the argmax of $\hat{\mathbf{y}} - A^\top\lambda$) leads to constraint violation, targeting updating cases where the current duals are misleading.

2.2 Regret Guarantees

Let $x^{\mathrm{IP}}(\hat{y})$ and $x^{\mathrm{LP}}(\hat{y})$ denote the solutions to Problem 1 and its LP relaxation, respectively, and let $\tilde{z}_{g,\tau}(\hat{y}, \lambda)$ and z_g^* be as defined above. Our analysis relies on three assumptions: (A1) the LP relaxation is integral, so $x^{\mathrm{IP}}(\hat{y}) = x^{\mathrm{LP}}(\hat{y})$; (A2) group maximizers are unique with a margin $\gamma > 0$, which ensures $\tilde{z}_{g,0} = z_g^*$; and (A3) rewards are bounded almost surely (full statements in Appendix).

Define the per-sample and expected losses as $\ell_\tau(\theta; \mathbf{x}, \mathbf{y}) := -\frac{1}{|\mathcal{G}|} \sum_{g \in \mathcal{G}} \big(\mathbf{y}_i\big)_{i \in g}^\top \tilde{z}_{g,\tau}\big(M_\theta(\mathbf{x}_i), \hat{\lambda}\big)$, and $\mathcal{L}_\tau(\theta) := \mathbb{E}[\ell_\tau(\theta; \mathbf{x}, \mathbf{y})]$, respectively, where $\hat{\lambda}$ is any dual-optimal vector for $\hat{\mathbf{y}} = M_\theta(\mathbf{x}_i)$. Let $\mathcal{L}_\tau^* := \inf_\theta \mathcal{L}_\tau(\theta)$ denote the Bayes risk. At $\tau = 0$,

$$\mathcal{L}_0^* = -\frac{1}{|\mathcal{G}|}\, \mathbb{E}\big[\, \mathbf{y}^\top x^{\mathrm{IP}}(\mathbf{y}) \,\big] = -\frac{1}{|\mathcal{G}|}\, \mathbb{E}\big[\, \mathbf{y}^\top x^{\mathrm{LP}}(\mathbf{y}) \,\big]\,,$$

where the second equality uses (A1), and the first holds because the infimum over predictors is achieved by choosing $x^{\mathrm{IP}}(Y)$. In DFL, the performance of a predictive model with parameters θ is measured by the decision regret: $\mathrm{Regret}(\theta) := \mathbb{E}\left[\mathbf{y}^\top x^{\mathrm{IP}}(\mathbf{y}) - \mathbf{y}^\top x^{\mathrm{IP}}(M_\theta(\mathbf{x}))\right]$.

The following result shows that minimizing DGL during training yields predictors with low decision regret; the proof is deferred to the Appendix.

Theorem 1. *Assume* (A1)–(A3). *Then for every $\tau > 0$ and $\theta \in \Theta$,* $\mathrm{Regret}(\theta) \;\le\; |\mathcal{G}| \left[\mathcal{L}_\tau(\theta) - \mathcal{L}_\tau^*\right] \;+\; O\big(e^{-\gamma/\tau}\big)$, *where the $O\big(e^{-\gamma/\tau}\big)$ term is uniform in θ (its constant may depend on C and $|\mathcal{G}|$). Consequently, if $\mathcal{L}_{\tau_k}(\theta_k) - \mathcal{L}_{\tau_k}^* \to 0$ and $\tau_k \downarrow 0$, then* $\mathrm{Regret}(\theta_k) \to 0$. *Since $e^{-\gamma/\tau} = o(\tau)$ as $\tau \downarrow 0$, the remainder is $O(\tau)$ for sufficiently small τ.*

2.3 Time Complexity

We summarize the state-of-the-art DFL baselines and their per-epoch complexity under gradient-based training. With N instances, per-instance times are: T_M for a forward–backward pass of M_θ, T_O to solve the downstream optimization, and T_O' to differentiate through the optimization layer. When an offline surrogate is used, K is the sample count and T_{LODL} its training time.

Two-Stage trains M_θ with a supervised loss (e.g., MSE) against ground-truth costs. Its per-epoch time is dominated by network training, yielding $\Theta(NT_M)$.

SPO+ replaces regret with a closed-form, piecewise-linear surrogate built from optimal and perturbed solutions [Elmachtoub and Grigas, 2022]. In common DFL (LP downstream), each instance solves the LP once to get the primal for the subgradient; gradients pass through the surrogate, not the solver, resulting in complexity $\Theta!\big(N(T_M + T_O)\big)$, with T_O the LP solve time.

QPTL embed a differentiable convex layer (e.g., CVXPYLayer) [Wilder et al., 2019]. The linear objective is augmented with a quadratic term so the forward pass solves a QP whose KKT system is then reused for implicit differentiation. Per-epoch cost is $\Theta(N(T_M + T_O + T_O'))$, typically dominated by T_O' due to factorization, linear solves, and autograd overhead.

LODLs learn an offline regret surrogate from K samples and solver outputs, so M_θ can train without solving the downstream problem [Shah et al., 2022]. Per-epoch cost is $\Theta(NT_M)$ (as in two-stage); the one-time offline cost is $\Theta(KNT_O + KNT_{\mathrm{LODL}})$, amortizing solver calls but adding upfront cost and approximation error.

DGL (Ours) shapes a differentiable surrogate with fixed or periodically refreshed duals, avoiding backprop through the solver. With refresh period U, the per-epoch cost is $\Theta(NT_M + \frac{N}{U}T_O)$ (one solve per instance every U epochs). Setting $U = \infty$ recovers two-stage runtime while retaining decision-aware guidance from the initial duals; if only a fraction ρ of instances are auto-updated (e.g., with constraint violations), replace N/U with ρN.

In summary, SPO+ and QPTL keep the solver in the training loop for every instance (QPTL additionally differentiates through it); LODLs shift most cost to an offline surrogate-fitting phase; and DGL decouples solver frequency from gradient steps, interpolating between two-stage speed and fully decision-aware training via U (or ρ).

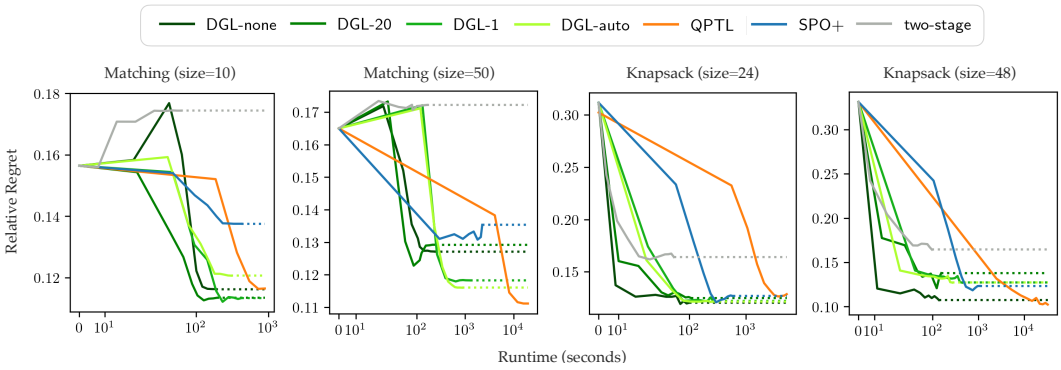

Figure 1: **Test relative regret vs. training time.** Across Matching (sizes 10, 50) and Knapsack (sizes 24, 48), DGL variants reach competitive or better regret far faster than QPTL and SPO+.

3 Experiments

We evaluate DGL on two PtO tasks: (i) many-to-one matching inspired by outcome-driven matching for refugee assignment [Bansak and Paulson, 2024], and (ii) weighted knapsack, a widely used benchmark in DFL [Mandi et al., 2024]. Details about the data generation process and training procedures for these two settings are provided in the Appendix. Because our experiments operate in finite-data, constrained regimes, we adopt a practical DGL variant that scores surrogate decisions under dual-adjusted reward. Specifically, we swap $\mathbf{y}^\top$ in $\mathcal{L}_\tau$ (Eq. 3) for $h_i(\mathbf{y}, \lambda))_{i \in g}$ to stabilize the learning and improve empirical performance. As baselines, we include Two-Stage, SPO+, and QPTL with consistent model architectures and solvers across methods[1].

Across all settings, DGL variants (no-update, fixed-frequency, auto-update) achieve competitive or lower regret *far sooner* than solver-in-the-loop baselines; the gap widens on larger instances, reflecting the growing cost of repeated solves. Concretely, the DGL curves in Fig. 1 drop rapidly at the start of training, while SPO+ and QPTL require substantially more time to reach similar regret levels. Light-touch refresh policies preserve this advantage: because DGL decouples solver frequency from gradient steps, fixed-frequency updates approach two-stage runtime when $U = \infty$ (no updates) yet remain decision-aware via duals. For a concrete reference point, on the larger settings (e.g., matching with 50 units and knapsack with 48 items), QPTL ultimately attains the lowest regret, but only by a modest margin and at a steep compute cost: reaching that level requires roughly $100\times$ more wall-clock time than DGL needs to achieve comparable regret. For equal time budgets, DGL delivers substantially lower regret. In practice, QPTL can be warm started with DGL by training first with DGL to quickly reach a strong decision-aware solution and switching to QPTL only when the marginal gains warrant the extra compute.

4 Discussion

Dual-guided learning preserves decision awareness while sharply reducing solver cost. Beyond efficiency, dual variables act as interpretable shadow prices that indicate which constraints drive the decision and by how much, offering an explanatory handle that is largely missing in black-box DFL methods based on differentiable layers or tight surrogates. Promising directions include formalizing constraint-attribution metrics that quantify each constraint's contribution to the decision or regret, and conducting sensitivity and stability analyses to track how shadow prices evolve during training and across instances. On the empirical side, we plan to extend experiments to larger scales and additional domains.

[1]We exclude LODLs to keep the evaluation focused on in-loop training; fair LODL comparisons require training an offline surrogate, which adds substantial upfront runtime that confounds per-epoch timing and is orthogonal to our contribution.

References

Brandon Amos and J. Zico Kolter. OptNet: Differentiable Optimization as a Layer in Neural Networks. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

Kirk Bansak and Elisabeth Paulson. Outcome-Driven Dynamic Refugee Assignment with Allocation Balancing. *Operations Research*, 72(6):2375–2390, 2024. ISSN 1526-5463.

Adam N. Elmachtoub and Paul Grigas. Smart "Predict, then Optimize". *Management Science*, 68(1): 9–26, 2022.

Jayanta Mandi, Emir Demirovi, Peter J. Stuckey, and Tias Guns. Smart Predict-and-Optimize for Hard Combinatorial Optimization Problems. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

Jayanta Mandi, James Kotary, Senne Berden, Maxime Mulamba, Victor Bucarey, Tias Guns, and Ferdinando Fioretto. Decision-Focused Learning: Foundations, State of the Art, Benchmark and Future Opportunities. *Journal of Artificial Intelligence Research*, 2024.

Sanket Shah, Kai Wang, Bryan Wilder, Andrew Perrault, and Milind Tambe. Decision-Focused Learning without Differentiable Optimization: Learning Locally Optimized Decision Losses. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, 2022.

Kai Wang, Shresth Verma, Aditya Mate, Sanket Shah, Aparna Taneja, Neha Madhiwalla, Aparna Hegde, and Milind Tambe. Scalable decision-focused learning in restless multi-armed bandits with application to maternal and child health. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence*, 2023.

Bryan Wilder, Bistra Dilkina, and Milind Tambe. Melding the data-decisions pipeline: decision-focused learning for combinatorial optimization. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*, 2019.

Appendix

A "Pick-one" group problem formulations

A.1 Assignment Problem

Consider the problem of assigning m workers to k jobs, where each job has capacity c_k. The match quality w_{lj} between a worker and job must be estimated.

The assignment problem is typically written as follows:

$$\max_{z\in\{0,1\}^{m\times k}} \sum_{l,j} w_{lj} z_{lj} \quad \text{s.t. } \sum_j z_{lj} = 1,\ \sum_l z_{lj} \le c_j.$$

To map this formulation to the Problem 1, consider vectorizing the decision variable $\mathbf{z}$:

$$x \triangleq (z_{11},\dots,z_{1k},\ z_{21},\dots,z_{mk})^\top \in \{0,1\}^n, \quad n = mk,$$

and similarly let $\mathbf{y}$ be the vectorized version of $\mathbf{w}$.

Each *worker row* is a group, since each worker can only be assigned to one job:

$$g_l = \big\{(l-1)k+1, ..., lk\big\}, \qquad l = 1,\dots,m.$$

The global constraint $Ax \le b$ must be constructed to reflect the capacity constraints $\sum_l z_{lj} \le c_j$. Therefore, $A \in \mathbb{R}^{k\times mk}$ has $A_{j,(l-1)k+j} = 1$ for $j = 1, ..., k$ and $l = 1, ..., m$ and is otherwise equal to zero. Additionally, $b = (c_1, ..., c_k)^\top$.

Now, notice that $(A^\top\lambda)_{j\in g_l} = (\lambda_1, ..., \lambda_k)$ and therefore $h_i(y,\lambda) = y_i - \lambda_i$.

A.2 Weighted Knapsack Problem

The classic weighted knapsack problem consists of items $i \in \{1,\dots,m\}$ with value v_i and weight w_i. To formulate this as above, we must be explicit about the item groups. Each item has two options: add to the knapsack or do not. Thus, each item will be in a group of size two.

The knapsack problem can be written as:

$$\max_{x\in\{0,1\}^{m\times 2}} \sum_i v_i x_{i1} \quad \text{s.t. } \sum_i w_i x_{i1} \le B \text{ for } j = 1,2, \quad \sum_{j=1,2} x_{ij} = 1 \text{ for all } i,$$

where $x_{i1} = 1$ if item i is added to the knapsack and $x_{i2} = 1$ if it is not. Additionally, B is the knapsack's capacity. As with the Assignment Problem, to map this problem to the general approach described above we will flatten the decision variables into a vector z, with $y = (v_1, 0, v_2, 0, ..., v_m, 0)$ and $A = (w_1, 0, w_2, 0, ..., w_m, 0)$. We also have that $g_l = \{2(l-1)+1, 2l\}$.

We can then show that $(h_i(\mathbf{y},\lambda))_{i\in g_l} = (y_{2(l-1)+1} - w_{2(l-1)+1}\lambda_1, 0)$. This is because for any group (i.e., item) l, the second option of *not* adding the item to the knapsack has a value of zero and a dual variable of zero. In other words, we should add item i to the knapsack if $y_i - w_i\lambda_1 \ge 0$.

A.3 Shortest Path Problem

Consider a directed graph $G = (V, E)$ with $|V| = N$ nodes and $|E| = M$ arcs. Let $s \in V$ be the *source* and $t \in V$ the *sink*. Each arc $e = (u, v) \in E$ has an (unknown) cost c_e that must be estimated; we work with *rewards* $y_e \triangleq -c_e$ so that a shortest s–t path maximizes the total reward. To allow a node to be *skipped*, attach a zero-reward dummy arc $e_v^{\text{skip}} = (v, \perp)$ leaving each non-sink node $v \ne t$. Denote the augmented arc set by $\bar{E}$ and its size by M^*.

The binary variables are:

$$x_e = \begin{cases} 1 & \text{if arc } e \text{ is used in the path,} \\ 0 & \text{otherwise,} \end{cases} \qquad e \in \bar{E}.$$

Index the arcs arbitrarily as $\bar{E} = \{e_1, \ldots, e_{M^*}\}$ and set

$$x \triangleq (x_{e_1}, \ldots, x_{e_{M^*}})^\top \in \{0,1\}^{M^*}, \quad y \triangleq (y_{e_1}, \ldots, y_{e_{M^*}})^\top.$$

For every non-sink node $v \neq t$ the set of outgoing arcs—including the dummy arc—forms a group (i.e., there must be exactly one outgoing arc from each node).

$$g_v \;=\; \{\, e = (v,u) \in \bar{E} \,\}, \qquad \sum_{e \in g_v} x_e = 1.$$

Hence $\mathcal{G} = \{\, g_v : v \in V \setminus t\}$. Let $A \in \{-1,0,1\}^{N \times M^*}$ be the node–arc incidence matrix

$$A_{ve} = \begin{cases} +1 & \text{if } e = (v,\cdot) \text{ leaves } v, \\ -1 & \text{if } e = (\cdot, v) \text{ enters } v, \\ 0 & \text{otherwise,} \end{cases}$$

Notice that dummy can leave nodes but not enter nodes. Define the supply vector

$$b_v = \begin{cases} +1 & v = s, \\ -1 & v = t, \\ 0 & v \in V \setminus \{s,t\}. \end{cases}$$

The equalities $Ax = b$ guarantee one unit of flow from s to t and zero net flow elsewhere.

For an arc $e = (u,v)$ we have

$$(A^\top \lambda)_e = \lambda_u - \lambda_v, \qquad h_e(y,\lambda) = y_e - \lambda_u + \lambda_v.$$

B Assumptions.

A1. **Integral polytope.** The feasible region of the LP relaxation of (BLP) is integral, hence its optimal solution x^{LP} is always binary and equals the integer optimum x^{IP}.

A2. **Unique group maximizer.** There exists $\gamma > 0$ such that for every group $g \in \mathcal{G}$ and for $c \in \{\, Y,\ \hat{y}(X;\theta)\,\}$,

$$\Delta_g(c) \;:=\; h_{j_g^\star}(c) \;-\; \max_{i \in g \setminus \{j_g^\star\}} h_i(c) \;\geq\; \gamma \qquad \text{almost surely,}$$

where $j_g^\star = \arg\max_{i \in g} h_i(c)$ and $h(c) = c - A^\top \lambda^\star(c)$ for any dual–optimal $\lambda^\star(c)$.

A3. **Bounded rewards.** $\|Y\|_\infty \leq C < \infty$ almost surely.

Assumption A1 implies that $x^{\mathrm{IP}}(\hat{y}) = x^{\mathrm{LP}}(\hat{y})$. We also note that under A2, $\tilde{z}_{g,0}$ reduces to z_g^*.

C Lemma 1

Lemma 1. *Fix any score vector $c \in \mathbb{R}^n$ and a group $g \in \mathcal{G}$. Let $j_g^\star = \arg\max_{i \in g} h_i(c)$ and define the* margin

$$\Delta_g(c) := h_{j_g^\star}(c) - \max_{i \in g \setminus \{j_g^\star\}} h_i(c) \;>\; 0 \quad \textit{a.s. by A2.}$$

Let $p_{g,\tau}(c)$ be the softmax probability assigned to $j_g^\star$:

$$p_{g,\tau}(c) = \frac{e^{h_{j_g^\star}(c)/\tau}}{\sum_{i \in g} e^{h_i(c)/\tau}} = \frac{1}{1 + \sum_{i \neq j_g^\star} e^{-(h_{j_g^\star}(c) - h_i(c))/\tau}} \;\geq\; 1 - \sum_{i \neq j_g^\star} e^{-(h_{j_g^\star}(c) - h_i(c))/\tau}.$$

Hence

$$1 - p_{g,\tau}(c) \;\leq\; \sum_{i \neq j_g^\star} e^{-(h_{j_g^\star}(c) - h_i(c))/\tau} \;\leq\; (|g| - 1)\, e^{-\Delta_g(c)/\tau}. \tag{4}$$

Moreover, the ℓ_1 distance between the one–hot vector $e_{j_g^\star}$ and the softmax vector on g satisfies

$$\left\| e_{j_g^\star} - \tilde{z}_{g,\tau}(c) \right\|_1 = (1 - p_{g,\tau}(c)) + \sum_{i \neq j_g^\star} p_{g,\tau,i}(c) = 2\,(1 - p_{g,\tau}(c)) \;\leq\; 2(|g| - 1)\, e^{-\Delta_g(c)/\tau}. \tag{5}$$

D Proof of Theorem 1

Proof. Let $\hat{x} := x^{\mathrm{IP}}(\hat{y})$ and $x^\star := x^{\mathrm{IP}}(Y)$. Add and subtract the softmax decision values at $\hat{y}$ and Y:

$$\begin{aligned}\mathrm{Regret}(\theta) &= \mathbb{E}\big[Y^\top x^\star - Y^\top \hat{x}\big] \\ &= \mathbb{E}\big[Y^\top x^\star - Y^\top \tilde{z}_\tau(Y)\big] - \mathbb{E}\big[-Y^\top \tilde{z}_\tau(Y)\big] \\ &\quad + \mathbb{E}\big[Y^\top \tilde{z}_\tau(\hat{y}) - Y^\top \hat{x}\big] + \mathbb{E}\big[-Y^\top \tilde{z}_\tau(\hat{y})\big].\end{aligned}$$

Recognizing $\mathbb{E}[-Y^\top \tilde{z}_\tau(\hat{y})] = |\mathcal{G}|\,\mathcal{L}_\tau(\theta)$ and $\mathcal{L}_\tau^* \le -\frac{1}{|\mathcal{G}|}\mathbb{E}[Y^\top \tilde{z}_\tau(Y)]$ (by plugging in Y for $\hat{y}$), we get

$$\mathrm{Regret}(\theta) \le |\mathcal{G}|\big[\mathcal{L}_\tau(\theta) - \mathcal{L}_\tau^*\big] + \mathbb{E}\big[Y^\top x^\star - Y^\top \tilde{z}_\tau(Y)\big] + \mathbb{E}\big[Y^\top \tilde{z}_\tau(\hat{y}) - Y^\top \hat{x}\big].$$

By Lemma 1 and A3,

$$\mathbb{E}\big[\big|Y^\top \hat{x} - Y^\top \tilde{z}_\tau(\hat{y})\big|\big] \le \delta_{\hat{y}}(\tau), \qquad \mathbb{E}\big[\big|Y^\top x^\star - Y^\top \tilde{z}_\tau(Y)\big|\big] \le \delta_Y(\tau),$$

where

$$\delta_{\hat{y}}(\tau) = 2C\,\mathbb{E}\Big[\sum_{g\in\mathcal{G}}(|g|-1)\,e^{-\Delta_g(\hat{y})/\tau}\Big], \qquad \delta_Y(\tau) = 2C\,\mathbb{E}\Big[\sum_{g\in\mathcal{G}}(|g|-1)\,e^{-\Delta_g(Y)/\tau}\Big].$$

Assumption A2 ensures $\Delta_g(\cdot) > 0$ almost surely, so $\delta_{\hat{y}}(\tau) \to 0$ and $\delta_Y(\tau) \to 0$ as $\tau \downarrow 0$.

Therefore, we have that

$$\mathrm{Regret}(\theta) \le |\mathcal{G}|\big[\mathcal{L}_\tau(\theta) - \mathcal{L}_\tau^*\big] + 2C\,\mathbb{E}\Big[\sum_{g\in\mathcal{G}}(|g|-1)\,e^{-\Delta_g(\hat{y})/\tau}\Big] + 2C\,\mathbb{E}\Big[\sum_{g\in\mathcal{G}}(|g|-1)\,e^{-\Delta_g(Y)/\tau}\Big]$$

Applying A2, we have that

$$\mathrm{Regret}(\theta) \le |\mathcal{G}|[\mathcal{L}_\tau(\theta) - \mathcal{L}_\tau^*] + O(e^{-\gamma/\tau}).$$

□