Task-agnostic Amortized Multi-Objective Optimization

Xinyu Zhang, Conor Hassan, **Julien Martinelli**, Daolang Huang, Samuel Kaski

ICLR 2026 submission ¯_(ツ)_/¯

October 30th, 2025

"Amortize everything 🤯"

— Daolang Huang

Black-box optimization

Let $\mathscr{X}$ be a design space and f an objective function, we seek

$$x = \operatorname*{argmax}_{x \in \mathscr{X}} f(x)$$

Black-box optimization

Let $\mathscr{X}$ be a design space and f an objective function, we seek

$$\mathrm{x} = \underset{\mathrm{x} \in \mathscr{X}}{\operatorname{argmax}} f(\mathrm{x})$$

- No information about f: gradients, convexity, invariances...
- Can only evaluate $y_i = f(\mathrm{x}_i) + \varepsilon$
- Collecting observations $\mathscr{D}_t = \{(\mathrm{x}_i, y_i)\}_{i=1}^{t}$

Black-box optimization

Let $\mathscr{X}$ be a design space and f an objective function, we seek

$$\mathrm{x} = \underset{\mathrm{x} \in \mathscr{X}}{\operatorname{argmax}} f(\mathrm{x})$$

- No information about f: gradients, convexity, invariances...
- Can only evaluate $y_i = f(\mathrm{x}_i) + \varepsilon$
- Collecting observations $\mathscr{D}_t = \{(\mathrm{x}_i, y_i)\}_{i=1}^{t}$

Bayesian Optimization tackles this problem by learning a *cheap-to-evaluate* **statistical surrogate** of f

$$f(\mathrm{x}) \sim \mathscr{GP}(\mu_{\theta_1}(\mathrm{x}), k_{\theta_2}(\mathrm{x}, \mathrm{x}'))$$

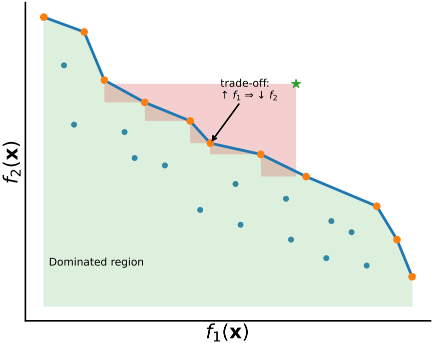
Yields next sample to query x_t selected sequentially based on an **acquisition function**

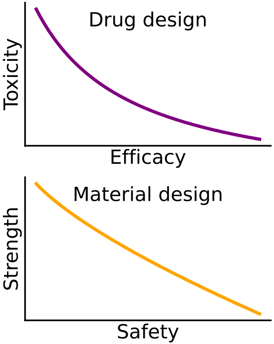
$$\mathrm{x}_{t+1} = \underset{\mathrm{x} \in \mathscr{X}}{\operatorname{argmax}} \alpha_{\theta_3}(\mathrm{x}|\mathscr{D}_t)$$

Evaluate $f(\mathrm{x}_{t+1})$; append $\mathscr{D}_{t+1} \leftarrow \mathscr{D}_t \cup \{(\mathrm{x}_{t+1}, y_{t+1})\}$; update surrogate $p(f|\mathscr{D}_{t+1})$, repeat until satisfied.

Multi-objective optimization

We now observe a *vector* of objectives $\mathrm{f}(\mathrm{x}) = [f_1(\mathrm{x}), \dots, f_M(\mathrm{x})] \in \mathbb{R}^M$





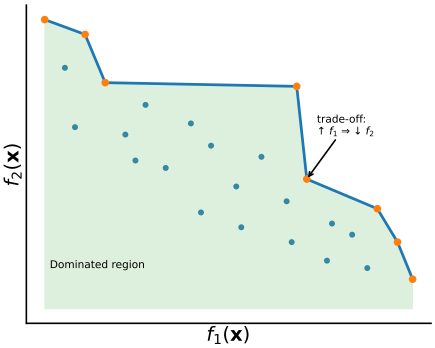
$\mathscr{P} \subset \mathbb{R}^m$ set of objective vectors and reference $\mathrm{r} \in \mathbb{R}^m$ with $\mathrm{r} \leq \mathrm{p}$ for all $\mathrm{p} \in \mathscr{P}$ (e.g., $r_i \leq \min_{\mathrm{p} \in \mathscr{P}} p_i$).

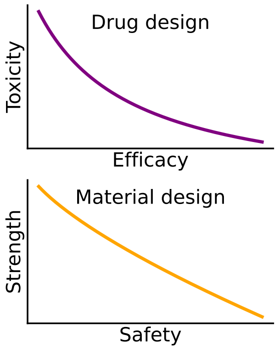
$$\mathrm{HV}(\mathscr{P}; \mathrm{r}) = \lambda_m\left(\bigcup_{\mathrm{p} \in \mathscr{P}} \prod_{i=1}^{m} [\, r_i, p_i \,]\right)$$

$$\mathrm{HVI}(\mathrm{x} \mid \mathscr{P}, \mathrm{r}) = \mathrm{HV}(\mathscr{P} \cup \{\mathrm{x}\}; \mathrm{r}) - \mathrm{HV}(\mathscr{P}; \mathrm{r})$$

Multi-objective optimization

We now observe a *vector* of objectives $\mathrm{f}(\mathrm{x}) = [f_1(\mathrm{x}), \ldots, f_M(\mathrm{x})] \in \mathbb{R}^M$





$\mathscr{P} \subset \mathbb{R}^m$ set of objective vectors and reference $\mathrm{r} \in \mathbb{R}^m$ with $\mathrm{r} \leq \mathrm{p}$ for all $\mathrm{p} \in \mathscr{P}$ (e.g., $r_i \leq \min_{\mathrm{p} \in \mathscr{P}} p_i$).

$$\mathrm{HV}(\mathscr{P}; \mathrm{r}) = \lambda_m\left(\bigcup_{\mathrm{p} \in \mathscr{P}} \prod_{i=1}^{m} [r_i, p_i]\right)$$

$$\mathrm{HVI}(\mathrm{x} \mid \mathscr{P}, \mathrm{r}) = \mathrm{HV}(\mathscr{P} \cup \{\mathrm{x}\}; \mathrm{r}) - \mathrm{HV}(\mathscr{P}; \mathrm{r})$$

Caveats

1. Performances highly dependent on surrogate and acquisition function pair
   - Requires careful, expert selection of the ideal combination
   - Will not transfer to the next problem

   → **"Learn" the language of optimization**

2. Slow in high-throughput settings due to vanilla GP cubic complexity
   → **Do a huge offline pre-training step, reduce inference to a feedforward pass**

3. Most of the time myopic, focused on 1-step optimality
   → **Train over long optimization horizons using RL**

4. Learning from previous campaigns not straightforward
   - What if I previously optimized $f_1(\mathrm{x})$ alone, does that help in optimizing $[f_1(\mathrm{x}), f_2(\mathrm{x})]$?
   - What if I previously optimized $\mathrm{f}(x_1, \dots, x_d)$, does that help in optimizing $\mathrm{f}(x_1, \dots, x_d, x_{d+1})$?

   → **Use dimension-agnostic task embeddings**

Caveats

1. Performances highly dependent on surrogate and acquisition function pair
   - Requires careful, expert selection of the ideal combination
   - Will not transfer to the next problem

   $\longrightarrow$ **"Learn" the language of optimization**

2. Slow in high-throughput settings due to vanilla GP cubic complexity

   $\longrightarrow$ **Do a huge offline pre-training step, reduce inference to a feedforward pass**

3. Most of the time myopic, focused on 1-step optimality

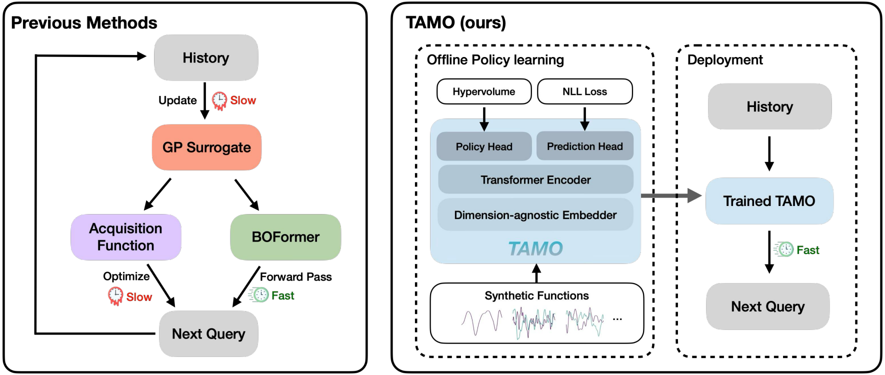
   $\longrightarrow$ **Train over long optimization horizons using RL**

4. Learning from previous campaigns not straightforward
   - What if I previously optimized $f_1(\mathrm{x})$ alone, does that help in optimizing $[f_1(\mathrm{x}), f_2(\mathrm{x})]$?
   - What if I previously optimized $\mathrm{f}(x_1, \dots, x_d)$, does that help in optimizing $\mathrm{f}(x_1, \dots, x_d, x_{d+1})$?

   $\longrightarrow$ **Use dimension-agnostic task embeddings**

Solution: Task-Agnostic Amortization 🤯

Overview

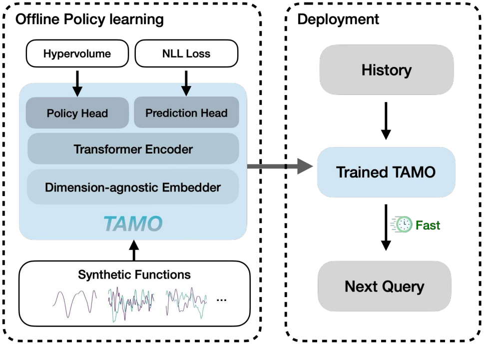


	Multi-objective	End-to-end amortized	Input agnostic	Output agnostic
Vanilla MOBO (Daulton et al., 2020)	✓	✗	N/A	N/A
BOFormer (Hung et al., 2025)	✓	✗	N/A	✗
NAP (Maraval et al., 2023)	✗	✓	✗	✗
DANP (Lee et al., 2025)	✗	✗	✓	✗
TAMO (this work)	✓	✓	✓	✓

TAMO: architecture at a glance

Shared backbone. Each forward pass uses either a *prediction* batch (*context, targets*) or an *optimization* batch (*history, queries*), traversing the *same core components*.
Optimization $\iff$ Predicting the optimum.

1. **Dimension-agnostic embedder:** scalar→vector maps producing tokens independent of input/output dim.
2. **Transformer encoder:** aggregates variable-size histories/contexts into a compact summary.
3. **Task conditioning:** a few learned tokens injected late to specialize the computation.
4. **Two heads:** a prediction head (density) and a policy head (acquisition over query set).

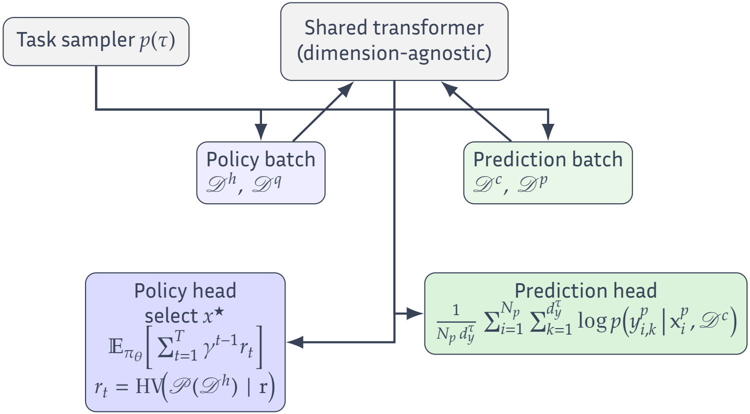


Pretraining

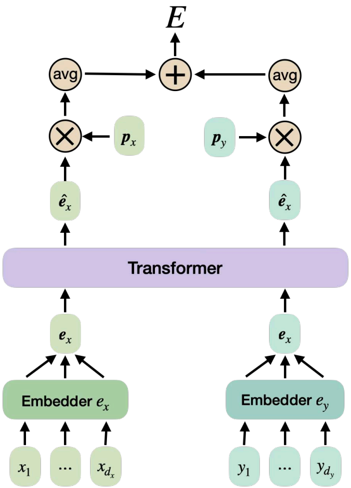
Task distribution. Synthetic MOO tasks $\tau \sim p(\tau)$ with $f_\tau : \mathscr{X} \subset \mathbb{R}^{d_x^\tau} \to \mathbb{R}^{d_y^\tau}$. Heterogeneous $d_x^\tau, d_y^\tau \Rightarrow$ *dimension-agnostic* policy.

Per step: two distinct mini-batches

- *Policy-learning:*
  history $\mathscr{D}^h = \{(\mathrm{x}^h, \mathrm{y}^h)\}_{h=1}^{N_h}$
  query set $\mathscr{D}^q = \{\mathrm{x}^q\}_{q=1}^{N_q}$
  policy picks the best x^q given $\mathscr{D}^h$.
- *Prediction:* fresh function draw; sample N pairs and split into context $\mathscr{D}^c = \{(\mathrm{x}^c, \mathrm{y}^c)\}_{c=1}^{N_c}$ and targets $\mathscr{D}^p = \{\mathrm{x}^p\}_{p=1}^{N_p}$ for in-context regression.



Dimension-agnostic embedder



- Learnable maps (NNs) $e_x, e_y : \mathbb{R} \to \mathbb{R}^{d_e}$ applied element-wise:
  $\mathrm{e}_x = e_x(\mathrm{x}) \in \mathbb{R}^{d_x^\tau \times d_e}, \quad \mathrm{e}_y = e_y(\mathrm{y}) \in \mathbb{R}^{d_y^\tau \times d_e}.$

- After L transformer layers over $[\mathrm{e}_x; \mathrm{e}_y]$ we get contextualized tokens $\hat{\mathrm{e}}_x, \hat{\mathrm{e}}_y$. Draw per-dimension tokens $\mathrm{p}_x^{(j)}, \mathrm{p}_y^{(k)} \in \mathbb{R}^{d_e}$ *sampled per batch from fixed pools of learned embeddings*, then:

$$\tilde{\mathrm{e}}_{x,j} = \hat{\mathrm{e}}_{x,j} \odot \mathrm{p}_x^{(j)}, \quad \tilde{\mathrm{e}}_{y,k} = \hat{\mathrm{e}}_{y,k} \odot \mathrm{p}_y^{(k)}$$

$$\bar{\mathrm{e}}_x = \frac{1}{d_x^\tau} \sum_{j=1}^{d_x^\tau} \tilde{\mathrm{e}}_{x,j}, \quad \bar{\mathrm{e}}_y = \frac{1}{d_y^\tau} \sum_{k=1}^{d_y^\tau} \tilde{\mathrm{e}}_{y,k}, \quad \mathrm{E} = \bar{\mathrm{e}}_x + \bar{\mathrm{e}}_y.$$

$\to$ breaks permutation symmetry; features/objectives remain identifiable.

Transformer encoder-decoder

Transformer layers split into $B = B_1 + B_2$

- B_1*:* history/context tokens self-attend $\Rightarrow \hat{\mathrm{E}}^h$ or $\hat{\mathrm{E}}^c$; queries/targets cross-attend to them $\Rightarrow \hat{\mathrm{E}}^q$ or $\hat{\mathrm{E}}^p$. This is the *only* path for queries/targets to use past data.
- B_2*:* drop history/context; keep query/target tokens + *task-specific tokens*. An attention mask enforces that query/target tokens only attend to task-specific tokens.

Task-specific tokens

- *Prediction:* a prediction-task token and an output-index token $\mathrm{p}_y^{(k)}$.
- *Optimization:* an optimization-task token, a time-budget token $g_{\text{time}} = \mathrm{MLP}_\theta\big((T-t)/T\big)$, and an input-dimension token $\sum_{j=1}^{d_x^\tau} \mathrm{p}_x^{(j)}$.

Heads

Prediction head (per scalar target):

$$\{\phi_{i\ell}, \mu_{i\ell}, \sigma_{i\ell}\}_{l=1}^{K} = \mathrm{MLP}_\theta(\hat{\mathrm{E}}_i^p)$$

$$p\big(y_{i,k}^p \mid \mathrm{x}_i^p, \mathscr{D}^c\big) = \sum_{\ell=1}^{K} \phi_{i\ell}\, \mathcal{N}\big(y_{i,k}^p;\, \mu_{i\ell}, \sigma_{i\ell}^2\big)$$

Policy head (over queries):

$$\alpha_i = \mathrm{MLP}_\theta(\hat{\mathrm{E}}_i^q)$$

$$\pi_\theta(\mathrm{x}_i^q) = \frac{e^{\alpha_i}}{\sum_r e^{\alpha_r}}$$

Inference

Algorithm S2 TAMO Test-Time Algorithm

Require: Pre-trained TAMO model, new task τ_{test}, query budget T, initial history set $\mathcal{D}_0^h := \{\boldsymbol{x}^h, y^h\}$ (with random samples if empty),

1: $\mathcal{D}^h \leftarrow \mathcal{D}_0^h$ ▷ Initialize the history set
2: $\mathcal{P} \leftarrow \{y^h\}$ ▷ Initialize the Pareto set
3: **for** $t = 1, \ldots, T$ **do**
4: $\quad \mathbf{x}_t \sim \pi_\theta(\cdot \mid \mathcal{D}^h, t, T)$ ▷ Sample the next query location
5: $\quad y_t \leftarrow \text{Evaluate}(\mathbf{x}_t, \tau_{\text{test}})$
6: $\quad \mathcal{D}^h \leftarrow \mathcal{D}^h \cup \{(\mathbf{x}_t, y_t)\}$ ▷ Update the history set
7: $\quad \mathcal{P} \leftarrow \mathcal{P} \cup \{y_t\}$ ▷ Update the Pareto set with the new observation
8: **end for**
9: **return** $\mathcal{D}^h, \mathcal{P}$

Pre-training dataset composition

- Input dimensionality $d_x \sim \mathscr{U}(\{1, 2\})$ and output dimensionality $d_y \sim \mathscr{U}(\{1, 2, 3\})$.
- For output correlations, with $p = 1/2$, either:
  - independent output dimensions are sampled
  - drawn from a multi-task GP, with task covariance defined as $k(i, j) = (\mathrm{C}\mathrm{C}^T + \mathrm{diag}(\mathrm{v}))_{i,j}, i, j \in \{1, \cdots, d_y\}$, with C is a low-rank matrix with rank $r \sim \mathscr{U}(\{1, \cdots, d_y\})$.
- Data kernel along each output dimension:
  - Equally sampled from RBF, Matérn-3/2, Matérn-5/2
  - Standard deviation $\sigma \sim U([0.1, 1.0])$
  - Lengthscale $\ell \sim \mathscr{N}(2/3, 0.5)$ truncated to $[0.1, 2.0]$.
- The sampled function values y were centered and normalized to lie within $[-1, 1]^{d_y}$.

Pre-training dataset composition

- Input dimensionality $d_x \sim \mathscr{U}(\{1, 2\})$ and output dimensionality $d_y \sim \mathscr{U}(\{1, 2, 3\})$.

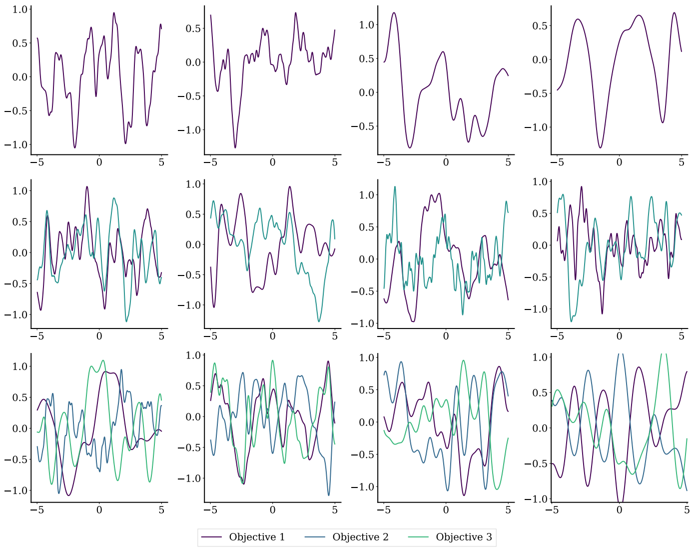
- For output correlations, with $p = 1/2$, either:
  - independent output dimensions are sampled
  - drawn from a multi-task GP, with task covariance defined as $k(i, j) = (\mathrm{C}\mathrm{C}^T + \mathrm{diag}(\mathrm{v}))_{i,j}, i, j \in \{1, \cdots, d_y\}$, with C is a low-rank matrix with rank $r \sim \mathscr{U}(\{1, \cdots, d_y\})$.

- Data kernel along each output dimension:
  - Equally sampled from RBF, Matérn-3/2, Matérn-5/2
  - Standard deviation $\sigma \sim U([0.1, 1.0])$
  - Lengthscale $\ell \sim \mathscr{N}(2/3, 0.5)$ truncated to $[0.1, 2.0]$.

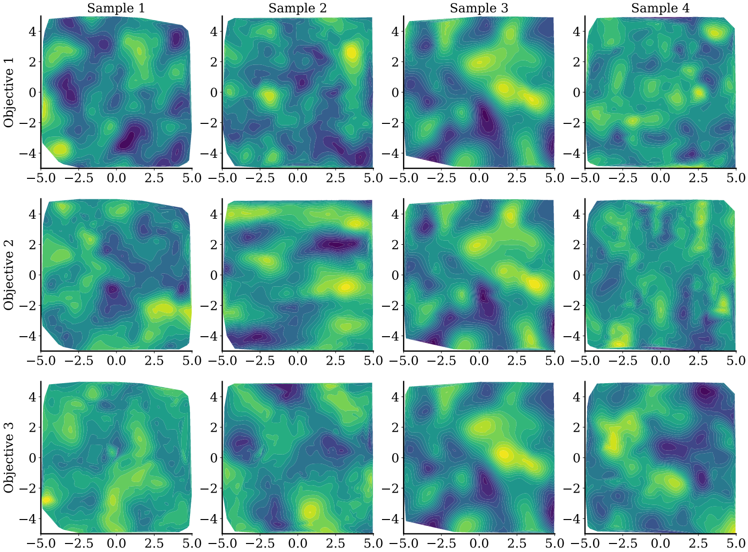
- The sampled function values y were centered and normalized to lie within $[-1, 1]^{d_y}$.

Completely synthetic dataset! 😲

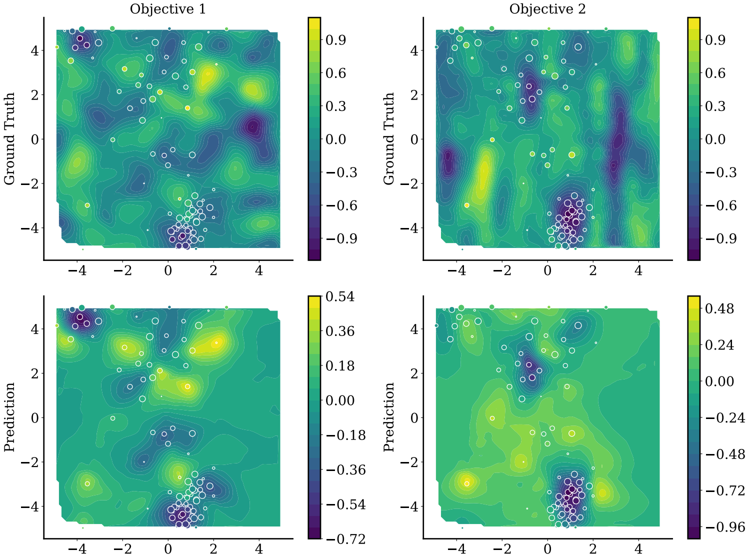
Samples from pre-training dataset, $d_x = 1, d_y = 3$



Samples from pre-training dataset, $d_x = 2, d_y = 3$



Optimization run example, $d_x = 2, d_y = 2$



Behind the scenes 😰

Dimension-Agnostic Embedder	
Number of learnable positional tokens for $\boldsymbol{x}$	4
Number of learnable positional tokens for y	3
Number of Transformer layers (L)	4
Dimension of $\mathbf{e}_x$ and $\mathbf{e}_y$	64
Transformer Encoder–Decoder	
Dimension of Transformer inputs	64
Point-wise feed-forward dimension of Transformer	256
Number of self-attention layers in Transformer (B)	8
Number of self-attention heads in Transformer	4
Heads	
Number of hidden layers in policy head	3
Number of components in GMM head (K)	20
Number of hidden layers in MLP for each GMM component	3
Training	
Number of iterations	400000
Number of burn-in iterations	393500
Initial learning rate for warm-up iterations (lr_1)	$1 \cdot 10^{-4}$
Initial Learning rate after warm-up (lr_2)	$4 \cdot 10^{-5}$
Learning rate scheduling	Linearly increase from 0 to lr_1 in the first 5% of total iterations; Cosine decay to 0 over total iterations
Size of prediction batch	32
Size of policy-learning batch	16
Weight on prediction loss (λ_{rl})	1.0
discount factor (γ)	1.0
Size of context set	$N_c \sim U[2, 50 \cdot d_x^\tau]$
Size of target set (N_t)	$300 - N_c$
Size of query set (N_q)	256
Optimization budget T	100
Noise level σ	0.0
Number of initial observations during pretraining	1
Evaluation	
Number of initial observations during test time	1
Noise level σ	0.0
Size of query set (N_q)	2048
Optimization budget (T)	100

- Positional tokens define max dimensionality the model can handle

Behind the scenes 😰

Dimension-Agnostic Embedder	
Number of learnable positional tokens for $\boldsymbol{x}$	4
Number of learnable positional tokens for y	3
Number of Transformer layers (L)	4
Dimension of $\mathbf{e}_x$ and $\mathbf{e}_y$	64
Transformer Encoder–Decoder	
Dimension of Transformer inputs	64
Point-wise feed-forward dimension of Transformer	256
Number of self-attention layers in Transformer (B)	8
Number of self-attention heads in Transformer	4
Heads	
Number of hidden layers in policy head	3
Number of components in GMM head (K)	20
Number of hidden layers in MLP for each GMM component	3
Training	
Number of iterations	400000
Number of burn-in iterations	393500
Initial learning rate for warm-up iterations (lr_1)	$1 \cdot 10^{-4}$
Initial Learning rate after warm-up (lr_2)	$4 \cdot 10^{-5}$
Learning rate scheduling	Linearly increase from 0 to lr_1 in the first 5% of total iterations; Cosine decay to 0 over total iterations
Size of prediction batch	32
Size of policy-learning batch	16
Weight on prediction loss (λ_{rl})	1.0
discount factor (γ)	1.0
Size of context set	$N_c \sim U[2, 50 \cdot d_x^{\tau}]$
Size of target set (N_t)	$300 - N_c$
Size of query set (N_q)	256
Optimization budget T	100
Noise level σ	0.0
Number of initial observations during pretraining	1
Evaluation	
Number of initial observations during test time	1
Noise level σ	0.0
Size of query set (N_q)	2048
Optimization budget (T)	100

- Positional tokens define max dimensionality the model can handle
- For comparison: TabPFN uses $L = 12$, $\mathrm{e}_x \in \mathbb{R}^{512}$ to handle $d_x \approx 50$

Behind the scenes 😰

Dimension-Agnostic Embedder	
Number of learnable positional tokens for $\boldsymbol{x}$	4
Number of learnable positional tokens for y	3
Number of Transformer layers (L)	4
Dimension of $\mathbf{e}_x$ and $\mathbf{e}_y$	64
Transformer Encoder–Decoder	
Dimension of Transformer inputs	64
Point-wise feed-forward dimension of Transformer	256
Number of self-attention layers in Transformer (B)	8
Number of self-attention heads in Transformer	4
Heads	
Number of hidden layers in policy head	3
Number of components in GMM head (K)	20
Number of hidden layers in MLP for each GMM component	3
Training	
Number of iterations	400000
Number of burn-in iterations	393500
Initial learning rate for warm-up iterations (lr_1)	$1 \cdot 10^{-4}$
Initial Learning rate after warm-up (lr_2)	$4 \cdot 10^{-5}$
Learning rate scheduling	Linearly increase from 0 to lr_1 in the first 5% of total iterations; Cosine decay to 0 over total iterations
Size of prediction batch	32
Size of policy-learning batch	16
Weight on prediction loss (λ_{rl})	1.0
discount factor (γ)	1.0
Size of context set	$N_c \sim U[2, 50 \cdot d_x^\tau]$
Size of target set (N_t)	$300 - N_c$
Size of query set (N_q)	256
Optimization budget T	100
Noise level σ	0.0
Number of initial observations during pretraining	1
Evaluation	
Number of initial observations during test time	1
Noise level σ	0.0
Size of query set (N_q)	2048
Optimization budget (T)	100

- Positional tokens define max dimensionality the model can handle
- For comparison: TabPFN uses $L = 12$, $\mathrm{e}_x \in \mathbb{R}^{512}$ to handle $d_x \approx 50$
- ≈ 1.1M parameters

Behind the scenes 😰

Dimension-Agnostic Embedder	
Number of learnable positional tokens for $\boldsymbol{x}$	4
Number of learnable positional tokens for y	3
Number of Transformer layers (L)	4
Dimension of $\mathbf{e}_x$ and $\mathbf{e}_y$	64
Transformer Encoder–Decoder	
Dimension of Transformer inputs	64
Point-wise feed-forward dimension of Transformer	256
Number of self-attention layers in Transformer (B)	8
Number of self-attention heads in Transformer	4
Heads	
Number of hidden layers in policy head	3
Number of components in GMM head (K)	20
Number of hidden layers in MLP for each GMM component	3
Training	
Number of iterations	400000
Number of burn-in iterations	393500
Initial learning rate for warm-up iterations (lr_1)	$1 \cdot 10^{-4}$
Initial Learning rate after warm-up (lr_2)	$4 \cdot 10^{-5}$
Learning rate scheduling	Linearly increase from 0 to lr_1 in the first 5% of total iterations; Cosine decay to 0 over total iterations
Size of prediction batch	32
Size of policy-learning batch	16
Weight on prediction loss (λ_{rl})	1.0
discount factor (γ)	1.0
Size of context set	$N_c \sim U[2, 50 \cdot d_x^\tau]$
Size of target set (N_t)	$300 - N_c$
Size of query set (N_q)	256
Optimization budget T	100
Noise level σ	0.0
Number of initial observations during pretraining	1
Evaluation	
Number of initial observations during test time	1
Noise level σ	0.0
Size of query set (N_q)	2048
Optimization budget (T)	100

- Positional tokens define max dimensionality the model can handle
- For comparison: TabPFN uses $L = 12$, $\mathrm{e}_x \in \mathbb{R}^{512}$ to handle $d_x \approx 50$
- $\approx$ 1.1M parameters
- Training time $\approx$ 2 days
- The model "learns" from 100-iterations tasks $\implies$ outside is OOD!

Behind the scenes 😰

Dimension-Agnostic Embedder	
Number of learnable positional tokens for $\boldsymbol{x}$	4
Number of learnable positional tokens for y	3
Number of Transformer layers (L)	4
Dimension of $\mathbf{e}_x$ and $\mathbf{e}_y$	64
Transformer Encoder–Decoder	
Dimension of Transformer inputs	64
Point-wise feed-forward dimension of Transformer	256
Number of self-attention layers in Transformer (B)	8
Number of self-attention heads in Transformer	4
Heads	
Number of hidden layers in policy head	3
Number of components in GMM head (K)	20
Number of hidden layers in MLP for each GMM component	3
Training	
Number of iterations	400000
Number of burn-in iterations	393500
Initial learning rate for warm-up iterations (lr_1)	$1 \cdot 10^{-4}$
Initial Learning rate after warm-up (lr_2)	$4 \cdot 10^{-5}$
Learning rate scheduling	Linearly increase from 0 to lr_1 in the first 5% of total iterations; Cosine decay to 0 over total iterations
Size of prediction batch	32
Size of policy-learning batch	16
Weight on prediction loss (λ_{rl})	1.0
discount factor (γ)	1.0
Size of context set	$N_c \sim U[2, 50 \cdot d_x^\tau]$
Size of target set (N_t)	$300 - N_c$
Size of query set (N_q)	256
Optimization budget T	100
Noise level σ	0.0
Number of initial observations during pretraining	1
Evaluation	
Number of initial observations during test time	1
Noise level σ	0.0
Size of query set (N_q)	2048
Optimization budget (T)	100

- Positional tokens define max dimensionality the model can handle
- For comparison: TabPFN uses $L = 12$, $\mathrm{e}_x \in \mathbb{R}^{512}$ to handle $d_x \approx 50$
- $\approx$ 1.1M parameters
- Training time $\approx$ 2 days
- The model "learns" from 100-iterations tasks $\implies$ outside is OOD!
- Query set = grid for evaluations! Increases inference time linearly.

Results

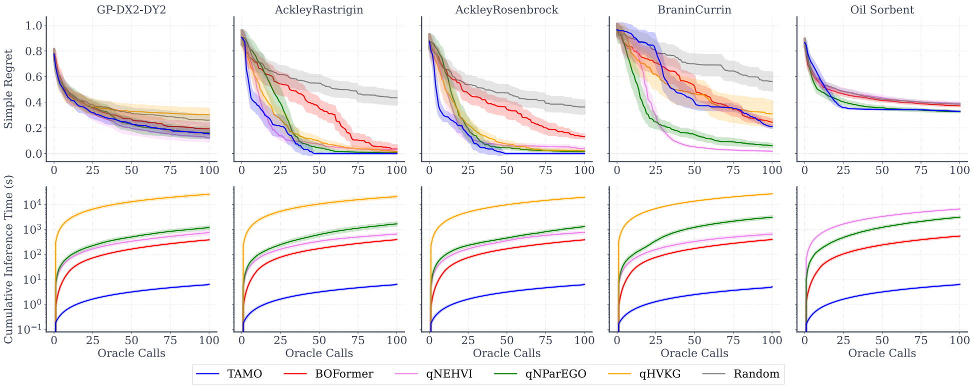


Figure 3: **Synthetic and real-world multi-objective benchmarks:** simple regret (top) and cumulative inference time (bottom) vs. oracle calls (mean $\pm$ 95% CIs over 30 runs). **TAMO achieves competitive regret while cutting proposal time by 50×–1000×.**

Results - Single-objective BO

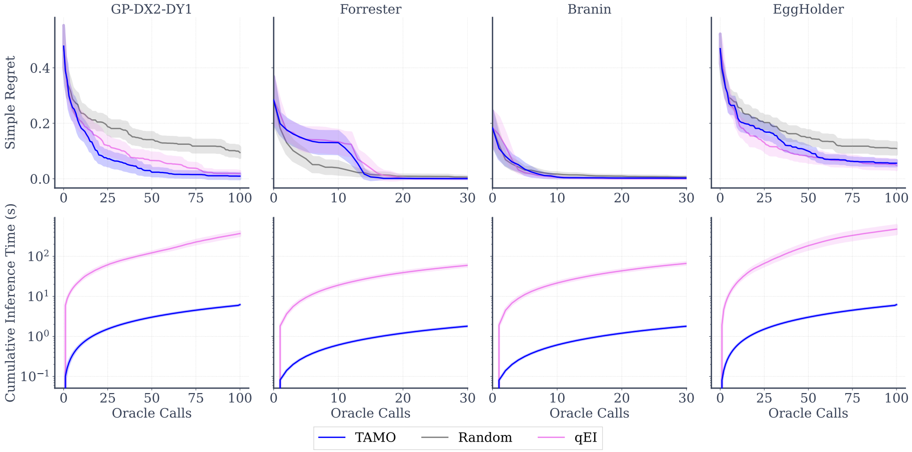


Figure S1: Simple regret and inference time on **synthetic examples for single-objective optimization**. Mean with 95% confidence intervals computed across 30 runs with random initial observations. **Again, TAMO matches state-of-the-art regret while dramatically reducing proposal time.**

Results - Out of distribution examples

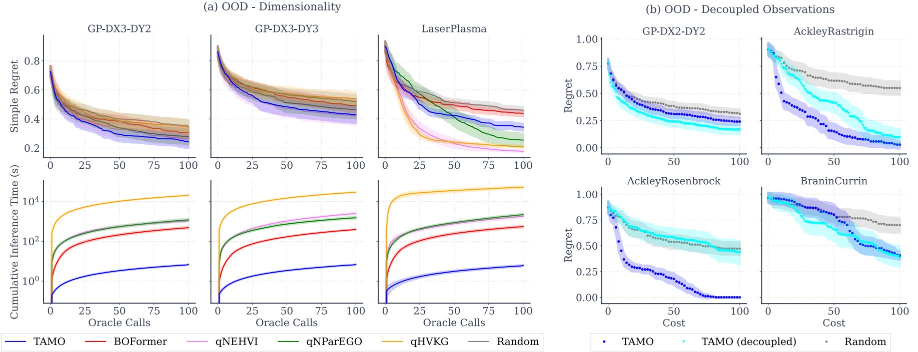


Figure 4: **Out-of-distribution evaluations. (a) Dimensionality:** simple regret (top) and cumulative inference time (bottom) on tasks whose input/output dimensions are unseen at pretraining. **(b) Decoupled observations:** regret vs. *cumulative cost* when, at step t, the optimizer may observe both objectives at cost 2 (dark blue) or only one at cost 1 (cyan). Curves show means with 95% confidence intervals over 30 runs with random initial observations. **TAMO offers promising generalization capabilities across unseen dimensionalities and decoupled feedback settings, delivering orders-of-magnitude faster proposals while maintaining competitive regret.**

Conclusion, Limitations, Perspectives

Universal black-box optimization model

- Extending to constrained, multi-source, preferential, cost-aware, etc...Does not require a paradigm shift, mostly pre-training modification
- Sky's the limit: just make the model bigger, increase pretraining dataset size, train longer

Conclusion, Limitations, Perspectives

Universal black-box optimization model

- Extending to constrained, multi-source, preferential, cost-aware, etc...Does not require a paradigm shift, mostly pre-training modification
- Sky's the limit: just make the model bigger, increase pretraining dataset size, train longer

Actually...

- The policy will quickly overfit 🤐

Conclusion, Limitations, Perspectives

Universal black-box optimization model

- Extending to constrained, multi-source, preferential, cost-aware, etc…Does not require a paradigm shift, mostly pre-training modification
- Sky's the limit: just make the model bigger, increase pretraining dataset size, train longer

Actually…

- The policy will quickly overfit 🤐
- Does not scale yet 🤬

Conclusion, Limitations, Perspectives

Universal black-box optimization model

- Extending to constrained, multi-source, preferential, cost-aware, etc...Does not require a paradigm shift, mostly pre-training modification
- Sky's the limit: just make the model bigger, increase pretraining dataset size, train longer

Actually...

- The policy will quickly overfit 🤐
- Does not scale yet 🤬
- Highly black box method 🫥

Conclusion, Limitations, Perspectives

Universal black-box optimization model

- Extending to constrained, multi-source, preferential, cost-aware, etc...Does not require a paradigm shift, mostly pre-training modification
- Sky's the limit: just make the model bigger, increase pretraining dataset size, train longer

Actually...

- The policy will quickly overfit 🤐
- Does not scale yet 🤬
- Highly black box method 🫥

Enough work avenues to fill several PhD theses

- Scaling to high-dimensional settings, handling structured objects
- Explainability
- Pre-training dataset $\iff$ Prior from a Bayesian perspective. Its composition is key.

Appendix 🤫

Two-steps training

1. Warm up backbone on prediction task by minimizing a negative log-likelihood over $(\mathscr{D}^c, \mathscr{D}^p)$

$$\mathscr{L}^{(p)}(\theta) = -\mathbb{E}_{\tau\sim p(\tau)}\left[\frac{1}{N_p\, d_y^\tau}\sum_{i=1}^{N_p}\sum_{k=1}^{d_y^\tau}\log p\left(y_{i,k}^p \,\middle|\, \mathrm{x}_i^p, \mathscr{D}^c\right)\right]$$

$\rightarrow$ promotes accurate in-context regression and useful representations

2. Then, optimize the policy $\pi_\theta(\mathrm{x} \mid s)$ with trajectory objective

$$J(\theta) \;=\; \mathbb{E}_{\tau\sim p(\tau)}\left[\mathbb{E}_{\pi_\theta}\left[\sum_{t=1}^{T}\gamma^{t-1} r_t\right]\right]$$

$\rightarrow$ aligns learning signal with improvements in Pareto quality alongside the prediction objective.

The overall objective combines both terms:

$$\mathscr{L}(\theta) = \mathscr{L}^{(p)}(\theta) + \lambda_{\mathrm{rl}}\mathscr{L}^{(\mathrm{rl})}(\theta), \qquad \mathscr{L}^{(\mathrm{rl})}(\theta) = -J(\theta),$$

- $\mathscr{L}^{(p)}$ and $\mathscr{L}^{(\mathrm{rl})}$ calculated from two distinct forward passes with different datasets
- Then summed for one single backward pass
- Training on full trajectories directly rewards long-horizon improvements,
- Amortization enables learning from many tasks offline.

Pre-training

Algorithm S1 TAMO Pre-Training Algorithm

Require: task distribution $p(\tau)$, prediction context size N_c, prediction target size N_p, query budget T, number of burn-in iterations η, number of total iterations num_total_iterations

for iteration $i = 1, \ldots,$ num_total_iterations **do**

▷ Prediction task

  Sample a task $\tau \sim p(\tau)$

  Sample prediction batches $\mathcal{D}^c = \{(\boldsymbol{x}_i^c, y_i^c)\}_{i=1}^{N_c}$ and $\mathcal{D}^p = \{\boldsymbol{x}_i^p\}_{i=1}^{N_p}$ from τ

  Model predicts outcomes: $p(y_{i,k}^p \mid \boldsymbol{x}_i^p, \mathcal{D}^c), \forall \boldsymbol{x}_i^p \in \mathcal{D}^p$

  if $i \leq \eta$ **then**

    Update model by minimizing the prediction loss $\mathcal{L}^{(p)}$ (Equation 5)

  else ▷ Policy learning task after burn-in phase

    Sample a new task $\tau \sim p(\tau)$

    Sample query set $\mathcal{D}^q$

    Initialize a history set $\mathcal{D}^h \leftarrow \{(\boldsymbol{x}_0^h, y_0^h)\}, \boldsymbol{x}_0^h \sim \mathcal{D}^q$

    Set reference point $\boldsymbol{r}$ and calculate optimal Hypervolume: $\text{HV}^* \leftarrow \text{HV}(\mathcal{P}(\mathcal{X}) \mid \mathbf{r})$

    Initialize Pareto set $\mathcal{P} \leftarrow \{y_0^h\}$

    for $t = 1, \ldots, T$ **do**

      Select next query point: $\boldsymbol{x}_t \sim \pi_\theta(\cdot \mid \mathcal{D}^h, t, T)$

      $y_t \leftarrow \text{Evaluate}(\boldsymbol{x}_t, \tau)$

      Update history set: $\mathcal{D}^h \leftarrow \mathcal{D}^h \cup \{(\boldsymbol{x}_t, y_t)\}$

      Update Pareto set: $\mathcal{P} \leftarrow \mathcal{P} \cup \{y_t\}$

      Compute reward: $r_t = \frac{\text{HV}(\mathcal{P}|\mathbf{r})}{\text{HV}^*}$

    end for

    Update model using the overall objective $\mathcal{L}$ (Equation 6)

  end if

end for