CAP: LEARNING TO OPTIMIZE FROM SPARSE OBSERVATIONS OF RELATED TASKS

Anonymous authors
Paper under double-blind review

ABSTRACT

In experiment-driven black-box optimization, each evaluation can consume materials, equipment time, and other resources, limiting the number of experiments available for a new task. Records from completed related optimization campaigns could help, but learning a sequential query policy remains challenging when task-specific experience consists of only a few sparse histories. We propose CAP (Candidate Acquisition Policy), which models shared structure and task-specific variation in fixed historical records to sample interactive practice functions. On these functions, CAP learns a feedback-conditioned query policy aimed at final optimization performance under a fixed evaluation budget. At deployment, its parameters remain fixed while each new observation informs the next query. Across five synthetic task families spanning 6 and 20 dimensions and two engineering simulation benchmarks, BOPTEST and SU2, CAP achieves the strongest final-budget performance among the evaluated optimizers. These results indicate that sparse records from completed related tasks can support adaptive query decisions on new tasks, making more effective use of limited, costly experimental evaluations.

1 INTRODUCTION

Experiment-driven black-box optimization seeks high-quality solutions with a limited number of objective evaluations. Each evaluation may consume materials or equipment time, or require a time-consuming engineering simulation, making effective use of every query on a new task important (Shields et al., 2021; Frazier, 2018). Bayesian optimization (BO) uses observations and predictive uncertainty to guide subsequent queries under such budgets (Frazier, 2018). Yet a new task need not start entirely from scratch: completed optimization campaigns on related tasks leave records that may contain useful experience. This paper asks how that experience can be turned into effective query decisions on a new task.

We study a setting with limited historical experience: only sparse optimization records from a small number of related tasks are available, and these completed tasks cannot be evaluated further. The goal is to learn a query policy from the records that uses incoming feedback to optimize a new, related task within a fixed evaluation budget. Related tasks can share useful structure without having identical response functions, so configurations that worked well in the past cannot simply be reused. A policy must draw on historical experience while continually revising its decisions in light of observations from the current task.

Transfer BO methods can use historical observations to improve predictions or query selection on a new task (Wistuba & Grabocka, 2021). Learned optimizers go further by learning query behavior from related tasks (Volpp et al., 2019; Hsieh et al., 2021). Training such behavior requires tasks or approximations that can answer newly proposed queries; these interactions need not be physical experiments. In our setting, however, task-family-specific experience is limited to sparse, fixed records of completed campaigns. The records cannot directly answer queries proposed during policy training, and the available observations do not uniquely determine the response surfaces of related tasks. The central challenge is therefore not merely to identify promising locations from historical data, but to construct grounded interactive practice within this information boundary and learn a sequential strategy that adapts to feedback on a new task.

We propose CAP (Candidate Acquisition Policy). CAP models structure shared across historical tasks alongside task-specific variation, then samples interactive practice functions from the fitted dis-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

tribution. These functions provide sequential optimization experience without further evaluations of the historical tasks. On them, CAP learns candidate-selection behavior aimed at final performance under a fixed evaluation budget. When deployed on a new related task, the policy parameters remain fixed: decisions use the growing observation context and candidate information from a frozen pretrained predictor, and each new evaluation informs the next query. Historical experience is thus distilled into reusable query behavior, while adaptation to a particular target task occurs through feedback rather than policy fine-tuning.

We evaluate CAP on five synthetic task families spanning 6 and 20 dimensions and on two engineering simulation benchmarks, building control with BOPTEST (Blum et al., 2021) and airfoil design with SU2 (Economon et al., 2016), against single-task, transfer, and learned optimizers. Across the five synthetic families and two engineering benchmarks, CAP achieves the best final-budget performance among the evaluated methods. These results indicate that sparse records from completed related tasks can support adaptive queries on new tasks, making more effective use of limited, costly evaluations.

Our main contributions are:

- We develop a way to construct interactive practice functions from sparse, fixed records of related tasks, enabling sequential query-policy training from completed campaigns.
- We introduce CAP, which combines record-derived practice with policy learning aimed at final performance under a fixed budget. At deployment, the policy adapts its decisions through the growing observation context without updating its parameters.
- We evaluate the complete optimizer across synthetic task families with different landscapes and dimensions and two engineering simulation benchmarks, demonstrating its final-budget optimization performance.

2 Related Work

Work on query efficiency under limited evaluation budgets either improves search and acquisition rules or uses experience from related tasks to learn predictions and decisions. In the former direction, TuRBO organizes search within local trust regions (Eriksson et al., 2019), while FunBO discovers reusable acquisition functions (Aglietti et al., 2024). CAP focuses on a more constrained setting in the latter direction: only fixed records from a small number of related tasks are available, yet the goal is to train a policy that selects queries using feedback from a new task. We discuss related work on cross-task transfer, query policy learning, and in-context prediction.

2.1 Transfer Learning for Bayesian Optimization

Transfer BO typically uses historical experience in the surrogate or acquisition decisions for a new task. RGPE weights GP models according to the ranking performance of historical models on current-task observations (Feurer et al., 2018); ABLR relates regression models across tasks through a shared representation (Perrone et al., 2018); and FSBO meta-learns a deep-kernel surrogate that adapts to new-task observations (Wistuba & Grabocka, 2021). At the acquisition level, TAF combines historical-task information with expected improvement on the new task (Wistuba et al., 2018). CAP also uses cross-task relatedness, but the historical task model generates practice functions for policy training rather than predicting candidate responses at deployment; training incorporates historical experience into the policy parameters.

2.2 Learning Query Policies for Black-Box Optimization

MetaBO trains a neural acquisition function with reinforcement learning (Volpp et al., 2019); FSAF connects acquisition functions to action values (Hsieh et al., 2021); and NAP jointly learns predictive representations and acquisition decisions (Maraval et al., 2023). OptFormer takes another route, learning query behavior from existing hyperparameter optimization trajectories (Chen et al., 2022).

Interactive training of the first three methods requires responses at points selected by the policy: MetaBO uses evaluable tasks or inexpensive approximations, FSAF uses sampleable functions, and NAP uses candidate responses available from source tasks. In our setting, only sparse records from

a small number of completed tasks are available; if a policy selects an unrecorded point, the records cannot provide feedback, so subsequent training trajectories cannot proceed as in the original setups. Thus, the limitation is not in these policy models themselves, but in the inability to meet their original training conditions directly from fixed records. CAP constructs queryable computational practice from the records, then learns a feedback-conditioned policy aimed at final performance under a fixed budget.

2.3 Learned Priors and In-Context Prediction

Prior-data fitted networks (PFNs) make predictions conditioned on an observation context (Müller et al., 2021); TabPFN applies this approach to prediction on small tabular datasets (Hollmann et al., 2025), and PFNs4BO studies its use in Bayesian optimization (Müller et al., 2023). CAP uses a frozen pretrained predictor to provide candidate features for the current task and trains the query policy separately. This predictor differs from the practice-function model fitted to historical records: the former supports current decisions, whereas the latter provides an interactive environment only during training.

3 Method

3.1 Problem Setting and Overview

We consider related black-box minimization tasks with comparable decision variables, mapped to a common domain $\mathcal{X} = [0,1]^d$. The available historical experience consists of fixed optimization records from K completed tasks,

$$\mathcal{D} = \{D_h\}_{h=1}^{K}, \qquad D_h = \{(x_{h,i}, y_{h,i})\}_{i=1}^{n_h}. \tag{1}$$

The tasks need not have identical response functions. Training may use these records and quantities derived from them, but cannot obtain further evaluations of the historical objectives or interact with a new target task.

On a new related task $f_\star$, an initial design of n_0 evaluations is followed by $H = B - n_0$ adaptive queries within a total budget B. Let O_t contain the initial observations and the first t adaptive evaluations, and let $b_t = \min_{(x,y)\in O_t} y$ be the best value observed so far. The objective is a small terminal value b_H; when a task optimum is available to the evaluator, it can also compute terminal simple regret. Neither the optimum nor unqueried target responses are available to the policy. Historical record length n_h and target budget B have distinct roles, even when an experimental protocol assigns them the same numerical scale.

CAP turns the fixed records into interactive computational practice and learns sequential query behavior over complete budgets. For a given historical archive and budget condition, training yields one policy whose parameters remain frozen when deployed on new tasks associated with that archive. The method is fitted and trained separately for different archives; a single set of weights is not assumed to transfer across unrelated task families. Adaptation to a new task comes from its growing observation context and the resulting query decisions, not from retraining the policy. Figure 1 summarizes this process.

3.2 History-Conditioned Practice Functions

Fixed records provide responses only at previously queried locations and cannot directly answer new queries proposed during policy training. To obtain complete sequential training trajectories, we fit an approximate, history-conditioned task distribution and sample

$$\widetilde{f} \sim \widehat{\mathcal{P}}_{\mathcal{D}}. \tag{2}$$

Each draw defines a function path fixed throughout an optimization episode. Interacting with it is computation derived from the records, not another evaluation of a historical or target objective.

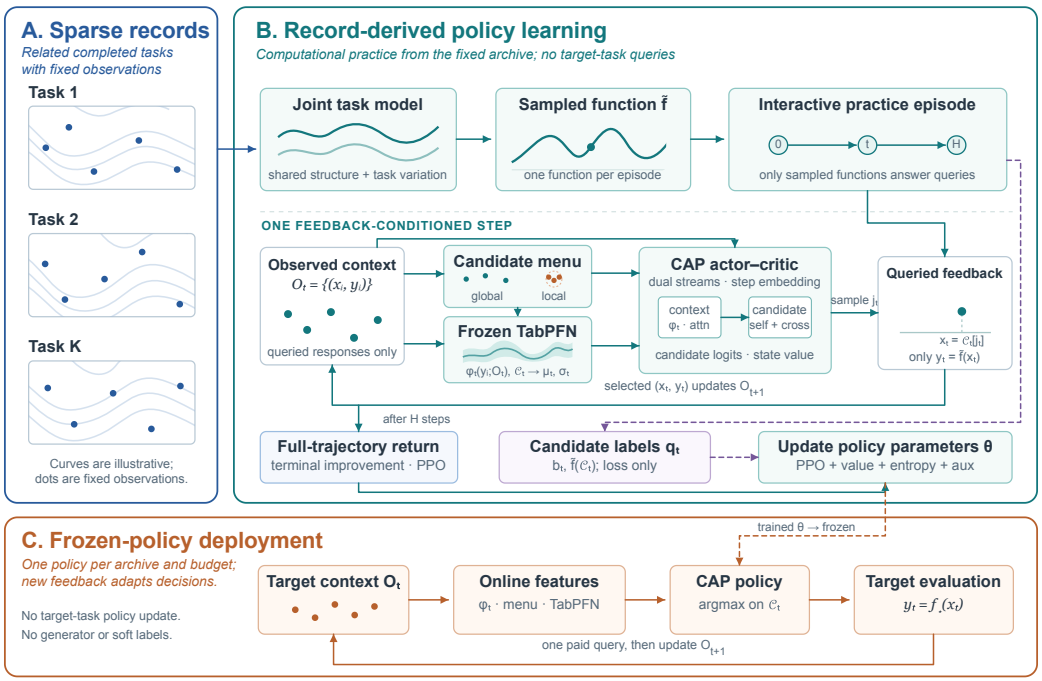

Figure 1: Overview of CAP. Fixed historical records fit an approximate distribution of coherent practice functions on which complete trajectories train a candidate-selection policy. At each step, the menu and represented observations inform a frozen, externally pretrained TabPFN and the actor–critic; only the selected response updates the observation context. Full trajectories yield PPO returns, while responses of all practice candidates supply auxiliary labels q_t only to the training loss. At deployment, one frozen policy per historical archive and budget condition uses current-task feedback and pays for only the selected target query.

Shared structure and task variation. For the continuous part of the responses, we model historical observations with shared spatial structure and task-specific variation:

$$
\begin{aligned}
z_{h,i} := T_\eta(y_{h,i}) &= \beta + g(x_{h,i}) + a_h + u_h(x_{h,i}) + \epsilon_{h,i}, \\
g \sim \mathcal{GP}(0, k_s), &\qquad u_h \overset{\text{iid}}{\sim} \mathcal{GP}(0, k_u), \\
a_h \overset{\text{iid}}{\sim} \mathcal{N}(0, v_a), &\qquad \epsilon_{h,i} \sim \mathcal{N}(0, \sigma_\epsilon^2).
\end{aligned} \tag{3}
$$

Here T_η transforms responses for historical modeling, β is a common mean, g captures structure shared across tasks, u_h captures task-specific spatial variation, and a_h is an optional task-level offset. The kernels k_s and k_u describe the shared and task-specific processes, respectively; $\epsilon_{h,i}$ is a fitting-noise term for historical observations. All K records contribute to the joint fit. In a new practice task, the shared structure is conditioned on the records, while its task-specific process and offset are drawn anew. A practice function therefore need not reproduce any particular historical response surface.

We approximate prior function paths with random Fourier features and condition whole paths on the historical observations (Rahimi & Recht, 2007; Wilson et al., 2020). The path and any applicable discrete response component are fixed at the start of an episode; repeated queries at the same location do not resample fitting noise. The resulting $\widehat{\mathcal{P}}_\mathcal{D}$ is an approximation constrained by sparse records, not an exact recovery of the true task distribution. Response-transformation families and whether a discrete component is enabled are specified for each experimental condition before target-task evaluation; their numerical parameters are fitted only from the corresponding historical records. These response-modeling choices do not change the subsequent policy architecture or training objective. Path construction and the response choices are detailed in Appendix A.

3.3 FEEDBACK-CONDITIONED CANDIDATE ACQUISITION

The historical archive influences query behavior through the trained policy parameters, but is not supplied as a direct input at each target-task decision. The policy instead uses the current task's evaluated observations, candidate locations, predictions based only on those observations, and the decision stage. The practice generator's latent variables and unqueried responses are unavailable.

Current-task response representation. To separate response modeling for practice-task generation from online decision features, define a context-dependent representation

$$\bar{y}_{t,i} = \phi_t(y_i; O_t), \qquad (x_i, y_i) \in O_t. \tag{4}$$

Unlike T_η, which is used to construct practice functions, ϕ_t depends only on the current task's evaluated observations and is recomputed after every query. The synthetic experiments use the identity representation; the engineering experiments use within-context normal scores based on response ranks, so wide response ranges do not compress input differences among favorable observations. Each representation is held fixed for its training condition and used identically at deployment; it is not selected using target-task outcomes. The representation affects only predictor and policy inputs. Incumbents, training returns, auxiliary labels, and reported objective values retain the original response units.

Candidate menu and online predictor. At step t, a fixed-size menu $\mathcal{C}_t = \{x_{t,j}\}_{j=1}^{M}$ combines a persistent global Sobol pool with local candidates regenerated around spatially separated, favorably observed locations. Selected global points are removed from the pool. Local sampling uses a predefined budget-dependent radius schedule, while its centers respond to the observations. The policy learns which available candidate to query; it does not learn the menu's allocation or radius rule.

A frozen, externally pretrained TabPFN-v2 regressor predicts candidate means $\mu_t(x)$ and standard deviations $\sigma_t(x)$ from $\{(x_i, \bar{y}_{t,i})\}_{(x_i,y_i)\in O_t}$ (Hollmann et al., 2025). These predictions are in the $\bar{y}$ representation space. TabPFN supplies current-task decision features, whereas the historical model in Eq. (3) supplies training environments. Task-family-specific practice and policy learning come from the fixed records, distinct from TabPFN's external pretraining.

Query decisions and feedback. A dual-stream attention actor–critic encodes observation and candidate tokens separately and uses candidate-to-context cross-attention to form feedback-conditioned candidate representations. A learned step embedding indicates the stage of the fixed query horizon. The actor assigns a logit $\ell_{\theta,t,j}$ to each candidate and defines

$$\pi_\theta(j \mid s_t) = \frac{\exp(\ell_{\theta,t,j})}{\sum_{k=1}^{M} \exp(\ell_{\theta,t,k})}, \tag{5}$$

where s_t comprises the represented observations, candidate features, and the stage within the fixed budget. The critic shares the encoder and estimates state value. Training samples an action from π_θ; deployment selects its highest-scoring candidate. Only that candidate's response is added to O_{t+1}, after which the representation, predictions, and menu are rebuilt. The policy can thus revise subsequent decisions in response to new feedback, but learns selection within the predefined menu rather than its allocation or radius rule.

3.4 LEARNING FROM TERMINAL RETURNS AND CANDIDATE RESPONSES

Practice functions provide two complementary training signals: terminal improvement over complete query trajectories and additional responses at the current candidates. The former targets fixed-budget optimization; the latter provides denser supervision at visited states, but reflects immediate response quality only.

Terminal objective. For each practice function $\widetilde{f}$, a fixed reference set determines a positive response scale $C_{\widetilde{f}}$ before the episode. The scale remains constant within the episode and is used only for training. With b_t computed from raw practice responses, define

$$r_t = \frac{b_t - b_{t+1}}{C_{\widetilde{f}}}, \qquad \sum_{t=0}^{H-1} r_t = \frac{b_0 - b_H}{C_{\widetilde{f}}}. \tag{6}$$

For a fixed practice function and initial design, the undiscounted return equals its scale-normalized terminal improvement; across functions, training optimizes the expectation of this quantity. No exact function optimum is needed for the reward.

We update the actor–critic with clipped PPO (Schulman et al., 2017), using returns and advantages from complete sampled trajectories. A query without immediate improvement can therefore receive a learning signal through later improvements in its trajectory. Optimization and numerical settings are given in Appendix A.

State-adaptive candidate supervision. At each visited state, the sampled practice function can also answer queries for the other candidates. Their signed, immediate utilities are

$$u_{t,j} = \frac{b_t - \widetilde{f}(x_{t,j})}{C_{\widetilde{f}}}. \tag{7}$$

We retain negative values so that candidates can still be ordered when none improves the incumbent. Sort utilities as $u_{t,(1)} \geq \cdots \geq u_{t,(M)}$ and form a soft label with a state-dependent temperature:

$$\begin{aligned} \tau_t &= \max\{\alpha(u_{t,(1)} - u_{t,(k)}), \epsilon_\tau\}, \\ q_{t,j} &= (1-\zeta)\frac{\exp(u_{t,j}/\tau_t)}{\sum_{\ell=1}^{M}\exp(u_{t,\ell}/\tau_t)} + \frac{\zeta}{M}. \end{aligned} \tag{8}$$

Here k is the candidate rank used to set the temperature, $\alpha > 0$ controls its scale, $\epsilon_\tau > 0$ supplies a numerical floor, and $\zeta \in [0, 1]$ mixes the labels with a uniform distribution. We detach q_t from gradient computation and minimize

$$\mathcal{L}_{\text{aux}} = -\mathbb{E}_{s_t}\sum_{j=1}^{M} q_{t,j}\log\pi_\theta(j \mid s_t). \tag{9}$$

These labels describe immediate response quality, not the long-term information value of a query. They complement, rather than replace, the complete-trajectory objective.

Joint training and deployment. The training loss combines the clipped PPO policy loss, a value loss, entropy regularization, and candidate supervision:

$$\mathcal{L} = \mathcal{L}_{\text{clip}} + c_V\mathcal{L}_V - c_{\text{ent}}\mathbb{E}_{s_t}\mathcal{H}(\pi_\theta(\cdot \mid s_t)) + \lambda_{\text{aux}}\mathcal{L}_{\text{aux}}. \tag{10}$$

Here $\mathcal{L}_{\text{clip}}$ is the PPO actor loss, $\mathcal{L}_V$ trains the critic to predict complete-trajectory returns, $\mathcal{H}$ denotes policy entropy, and $c_V, c_{\text{ent}}, \lambda_{\text{aux}}$ are loss weights. Responses of unselected candidates are used only to construct training labels; they do not enter the policy state, TabPFN context, or subsequent observation trajectory. These extra evaluations are computational queries to practice functions, not target-task evaluations.

After training, deployment retains the frozen predictor, candidate construction, and one fixed policy for the historical archive and budget condition. It does not use the practice generator or candidate labels. Each adaptive step pays for only the selected target query and returns the best point observed within the budget.

4 Experiments

We ask whether a policy learned from a small number of fixed historical optimization records can improve budget-limited optimization on new, related tasks. We first compare the complete optimizer across five synthetic task families with different landscapes and dimensions, then examine whether its performance extends to two engineering simulation benchmarks.

4.1 Experimental Setup

Tasks and budgets. We evaluate five synthetic families—Hartmann6, Ackley6, Levy6, Ackley20, and Levy20—each with 20 historical tasks ($5d$ fixed observations per task) and 100 distinct target tasks; their construction is outlined in Appendix B.2. BOPTEST building control and SU2 airfoil

design each have 20 historical and 20 target tasks (Blum et al., 2021; Economon et al., 2016). All methods use the same target tasks, shared initial design ($n_0 = d/2$), and total budget ($B = 5d$ evaluations) within each benchmark. CAP trains one policy per historical archive and freezes it on target tasks. Further protocol is in Appendix B; synthetic scoring is in Appendix C.

Baselines. Single-task optimizers are EI, PI, UCB, RolloutEI-2, FunBO, and TuRBO; FunBO uses a discovered acquisition function and TuRBO uses local trust regions (Aglietti et al., 2024; Eriksson et al., 2019). Record-based transfer methods are TAF-ranking, RGPE, ABLR, and FSBO (Wistuba et al., 2018; Feurer et al., 2018; Perrone et al., 2018; Wistuba & Grabocka, 2021). FSAF, MetaBO, and NAP learn query policies through interaction (Hsieh et al., 2021; Volpp et al., 2019; Maraval et al., 2023). We also include publicly pretrained PFNs4BO as an external-prior reference (Müller et al., 2023). Each retains its native surrogate, candidate construction, and decision rule; objective-evaluation budgets are matched.

FSAF, MetaBO, and NAP need training tasks that answer new queries, but the historical records cannot. We therefore train them on the same record-derived, queryable practice-function distribution as CAP, within the same historical-information boundary. They share practice feedback, not CAP's predictor, architecture, candidates, or loss. The distinct roles of CAP's frozen TabPFN predictor and the externally pretrained PFNs4BO model are explained in Section 3.3.

Evaluation metric. For synthetic tasks, we report median normalized simple regret across targets, at the final budget and along the trajectory. Since engineering optima are unknown, we report median best-observed objective values in original units. Lower is better. Appendix C details scoring and aggregation.

4.2 Synthetic Task Families

Table 1 compares final regret across all methods; Figure 2 shows trajectories for a fixed cross-category selection. All trajectories appear in Appendix C.

Table 1: Final median normalized simple regret on 100 target tasks per family (lower is better; best in bold). PFNs4BO supports at most 18 dimensions and is omitted in 20D.

Method	Hartmann6	Ackley6	Levy6	Ackley20	Levy20
CAP	**0.0166**	**0.0951**	**0.0182**	**0.1478**	**0.0986**
FSAF	0.7936	0.6786	0.0384	0.7808	0.1589
MetaBO	0.1441	0.6052	0.0721	0.7808	0.3311
NAP	0.2606	0.1672	0.0686	0.7407	0.3460
TAF-ranking	0.0925	0.3027	0.0922	0.7069	0.2611
RGPE	0.1404	0.3963	0.0793	0.7192	0.2778
ABLR	0.1035	0.4534	0.0426	0.7237	0.2414
FSBO	0.1157	0.2626	0.0911	0.6814	0.2310
PFNs4BO	0.2514	0.3380	0.1076	N/A	N/A
TuRBO	0.1698	0.3066	0.1044	0.4928	0.2165
EI	0.4762	0.5370	0.1384	0.7186	0.2904
PI	0.4314	0.4997	0.1364	0.7113	0.2824
UCB	0.5375	0.5459	0.1389	0.7205	0.2837
FunBO	0.4973	0.5265	0.1343	0.7142	0.2826
RolloutEI-2	0.5467	0.5213	0.1409	0.7238	0.2791

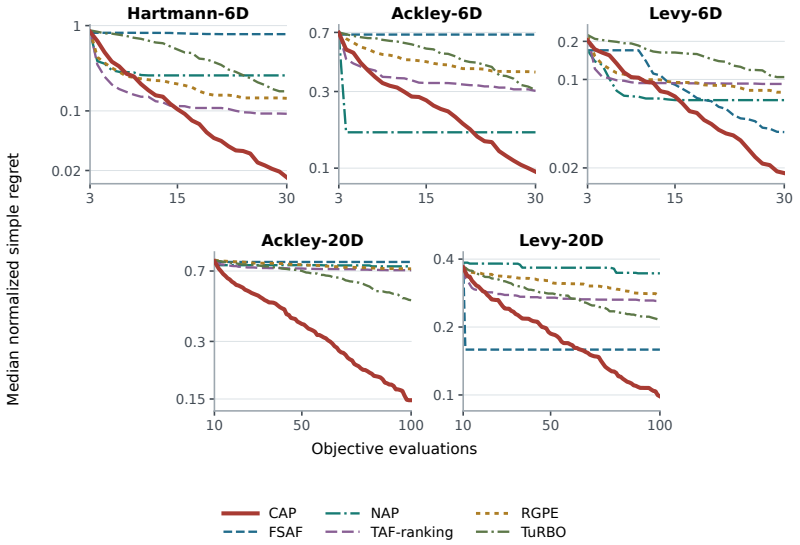

Figure 2: Median normalized simple regret across 100 target tasks per family (lower is better). Curves begin after the initial design, included in the evaluation count; panel scales differ. Appendix C shows all methods.

CAP has the lowest median final regret among the reported methods in all five families, including Ackley20 and Levy20 (Table 1). Its closest competitor differs across families, so the result is not confined to one landscape or baseline category. The trajectories in Figure 2 show how the methods improve with target-task feedback, complementing the fixed-budget comparison in the table.

4.3 Engineering Simulation Tasks

Synthetic families allow controlled variation between related tasks, whereas the engineering experiments use specific simulator objectives and operating conditions. BOPTEST varies operating time windows within a shared building and control space; SU2 varies flight conditions within a shared airfoil-design space. BOPTEST has $d = 6$ and a budget of 30 simulator evaluations per target task; SU2 has $d = 10$ and a budget of 50. Policy training uses only the fixed historical records and their derived practice tasks, not simulator calls on target tasks.

Table 2 reports the median final best-observed objective for all reported methods on both benchmarks. Figure 3 shows their progress within the simulation budget; trajectories for every reported method appear in Appendix D.

Table 2: Final median best-observed objective across 20 target tasks per engineering benchmark (lower is better). Values use each benchmark's original scale; bold marks the best result.

Method	BOPTEST (6D)	SU2 (10D)
CAP	**0.1677**	**0.0000405**
FSAF	0.1710	0.0023069
MetaBO	0.2078	0.0023150
NAP	0.1878	0.0027427
TAF-ranking	0.1773	0.0018606
RGPE	0.1926	0.0016985
ABLR	0.3900	0.0018256
FSBO	0.2786	0.0022342
PFNs4BO	0.1695	0.0028329
TuRBO	0.1801	0.0018902
EI	0.2057	0.0023928
PI	0.1943	0.0021505
UCB	0.1808	0.0021502
FunBO	0.1935	0.0020349
RolloutEI-2	0.1814	0.0020041

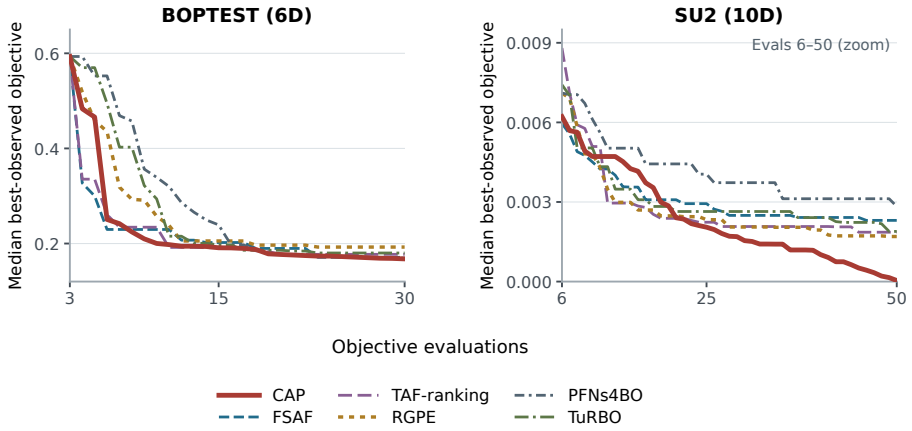

Figure 3: Median best-observed objective versus simulator evaluations (lower is better). Vertical axes use each benchmark's original scale. SU2 shows evaluations 6–50; its five initial evaluations count toward the budget. Appendix D shows all methods.

CAP obtains the lowest median final best-observed objective among the reported methods on both BOPTEST and SU2 (Table 2). The trajectories show how this performance develops across simulator evaluations (Figure 3). Together with the synthetic results, these experiments demonstrate its budget-limited optimization performance across the evaluated task families.

The two engineering benchmarks show different margins. On BOPTEST, the final medians of CAP (0.1677), PFNs4BO (0.1695), and FSAF (0.1710) are close. Several methods improve rapidly early in the budget and approach similar values near its end. On SU2, CAP reaches 0.0000405 after 50 evaluations, while the next lowest final median in Table 2 is RGPE's 0.0016985. Its median trajectory continues to decrease late in the budget, whereas the other displayed methods finish above it. These comparisons are within each benchmark: BOPTEST and SU2 retain their own objective scales, and the reported medians do not imply a win on every target task. The combined endpoint and trajectory results thus distinguish a narrow lead in BOPTEST from a larger final-budget separation in SU2.

5 Conclusion

We introduced CAP to turn fixed, sparse records of completed related tasks into interactive computational practice and learn a sequential query policy for a fixed evaluation budget. On a new task, the policy requires no further training: it adjusts subsequent queries as observations accumulate. Across five synthetic families and two engineering simulation benchmarks, CAP achieves the lowest median final-budget value among the compared methods. These results show that even sparse, non-extendable records can provide useful experience for feedback-conditioned optimization on new tasks, helping make better use of limited, costly objective evaluations.

This setting matters when past campaigns cannot be extended: the practice used for policy learning is generated computationally from their fixed records, while target-task evaluations are reserved for deployment. The policy then uses its learned parameters together with the observations collected on the current task, without updating those parameters between queries. Our evidence concerns new tasks related to the recorded ones and sharing their decision space; it does not establish performance on unrelated task families. Within that scope, the results support learning a reusable query policy from a small historical archive rather than requiring fresh interactions with the historical objectives.

AI USE STATEMENT

Generative AI tools were used to assist with language polishing of the manuscript, code optimization, and organization of figures and research materials. The authors reviewed the AI-assisted work and take responsibility for the final manuscript and research results.

REFERENCES

Virginia Aglietti, Ira Ktena, Jessica Schrouff, Eleni Sgouritsa, Francisco J. R. Ruiz, Alan Malek, Alexis Bellot, and Silvia Chiappa. Funbo: Discovering acquisition functions for bayesian optimization with funsearch, 2024. URL https://arxiv.org/abs/2406.04824.

David Blum, Javier Arroyo, Sen Huang, Ján Drgoňa, Filip Jorissen, Harald Taxt Walnum, Yan Chen, Kyle Benne, Draguna Vrabie, Michael Wetter, et al. Building optimization testing framework (boptest) for simulation-based benchmarking of control strategies in buildings. *Journal of Building Performance Simulation*, 14(5):586–610, 2021.

Yutian Chen, Xingyou Song, Chansoo Lee, Zi Wang, Qiuyi Zhang, David Dohan, Kazuya Kawakami, Greg Kochanski, Arnaud Doucet, Marc'aurelio Ranzato, Sagi Perel, and Nando de Freitas. Towards learning universal hyperparameter optimizers with transformers, 2022. URL https://arxiv.org/abs/2205.13320.

Thomas D Economon, Francisco Palacios, Sean R Copeland, Trent W Lukaczyk, and Juan J Alonso. Su2: An open-source suite for multiphysics simulation and design. *Aiaa Journal*, 54(3):828–846, 2016.

David Eriksson, Michael Pearce, Jacob Gardner, Ryan D Turner, and Matthias Poloczek. Scalable global optimization via local bayesian optimization. *Advances in neural information processing systems*, 32, 2019.

Matthias Feurer, Benjamin Letham, and Eytan Bakshy. Scalable meta-learning for bayesian optimization. *stat*, 1050(6):7, 2018.

Peter I Frazier. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 01 2025. doi: 10.1038/s41586-024-08328-6. URL https://www.nature.com/articles/s41586-024-08328-6.

Bing-Jing Hsieh, Ping-Chun Hsieh, and Xi Liu. Reinforced few-shot acquisition function learning for bayesian optimization. *Advances in Neural Information Processing Systems*, 34:7718–7731, 2021.

Alexandre Maraval, Matthieu Zimmer, Antoine Grosnit, and Haitham Bou Ammar. End-to-end meta-bayesian optimisation with transformer neural processes. *Advances in Neural Information Processing Systems*, 36:11246–11260, 2023.

Samuel Müller, Noah Hollmann, Sebastian Pineda Arango, Josif Grabocka, and Frank Hutter. Transformers can do bayesian inference. *arXiv preprint arXiv:2112.10510*, 2021.

Samuel Müller, Matthias Feurer, Noah Hollmann, and Frank Hutter. Pfns4bo: In-context learning for bayesian optimization. In *International Conference on Machine Learning*, pp. 25444–25470. PMLR, 2023.

Valerio Perrone, Rodolphe Jenatton, Matthias W Seeger, and Cédric Archambeau. Scalable hyperparameter transfer learning. *Advances in neural information processing systems*, 31, 2018.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20, 2007.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Benjamin J Shields, Jason Stevens, Jun Li, Marvin Parasram, Farhan Damani, Jesus I Martinez Alvarado, Jacob M Janey, Ryan P Adams, and Abigail G Doyle. Bayesian reaction optimization as a tool for chemical synthesis. *Nature*, 590(7844):89–96, 2021.

Michael Volpp, Lukas P Fröhlich, Kirsten Fischer, Andreas Doerr, Stefan Falkner, Frank Hutter, and Christian Daniel. Meta-learning acquisition functions for transfer learning in bayesian optimization. *arXiv preprint arXiv:1904.02642*, 2019.

James T. Wilson, Viacheslav Borovitskiy, Alexander Terenin, Peter Mostowsky, and Marc Peter Deisenroth. Efficiently sampling functions from gaussian process posteriors, 2020. URL `https://arxiv.org/abs/2002.09309`.

Martin Wistuba and Josif Grabocka. Few-shot bayesian optimization with deep kernel surrogates. *arXiv preprint arXiv:2101.07667*, 2021.

Martin Wistuba, Nicolas Schilling, and Lars Schmidt-Thieme. Scalable gaussian process-based transfer surrogates for hyperparameter optimization. *Machine Learning*, 107(1):43–78, 2018.

A METHOD IMPLEMENTATION

A.1 PRACTICE-FUNCTION CONSTRUCTION

Flatten the continuous-response historical observations and let h_i identify the task associated with record i. The model in Eq. (3) gives the covariance

$$C_{ij} = k_s(x_i, x_j) + \mathbf{1}[h_i = h_j]\{k_u(x_i, x_j) + v_a\} + \sigma_\epsilon^2 \mathbf{1}[i = j]. \tag{11}$$

Both spatial kernels are isotropic Matérn-$5/2$ kernels. A new task shares only g with historical tasks; its task-specific process $u_\star$ and offset $a_\star$ are sampled anew. Model parameters are fitted using the corresponding historical records, without choosing them from target-task responses or results.

We approximate prior paths with random Fourier features and condition whole paths on the historical observations. Let $z_\mathcal{D}$ be the transformed historical responses, $z_\mathcal{D}^0$ the prior draw at historical inputs (including the historical tasks' processes, offsets, and fitting noise), and $z_\star^0(x)$ the new task's prior latent path. With $c_s(x)_i = k_s(x, x_i)$, the continuous practice path, conditional on a mean and response-transform component, is

$$\widetilde{z}_\star(x) = \beta + z_\star^0(x) + c_s(x)^\top C^{-1} \left[z_\mathcal{D} - \beta \mathbf{1} - z_\mathcal{D}^0 \right]. \tag{12}$$

Applying the inverse response transformation, and any applicable fixed discrete response component, yields $\widetilde{f}(x)$. The path, transformation component, and discrete component remain fixed within a draw; historical fitting noise is not resampled on practice queries.

The joint model and path-conditioning procedure are shared across experiments. Response transformations and an optional discrete component accommodate different response distributions in the fixed historical records. Synthetic experiments use an affine transformation; engineering experiments use record-fitted monotone transformations, with a discrete component where repeated responses occur. These choices are fixed before target-task evaluation, with numerical parameters fitted only from the corresponding records; they do not alter the candidate policy, training objective, or deployment procedure.

A.2 ONLINE REPRESENTATION, CANDIDATES, AND TRAINING

The online representation ϕ_t in Eq. (4) serves decision inputs, whereas T_η serves practice-function construction. Synthetic runs use $\phi_t(y_i; O_t) = y_i$. In engineering runs, let $N_t = |O_t|$ and let r_i be the one-based average rank of y_i among the observed responses, with ties sharing their average rank. Then

$$\phi_t(y_i; O_t) = \Phi^{-1}\left(\frac{r_i - 0.5}{N_t} \right). \tag{13}$$

This representation is recomputed from current observations at every step, with the same rule used in training and deployment. It leaves raw-response incumbents, rewards, and evaluation unchanged. The wide range of historical engineering responses motivates representing their relative differences without introducing hidden task parameters.

Each decision uses $M = 256$ candidates: 128 points from an initial persistent scrambled Sobol pool and local points filling the remaining slots. Selected persistent points are removed. Local points are generated around up to three spatially separated, favorably observed locations. Their centers change with feedback, while the radius and allocation follow a predefined budget-dependent rule and are not policy actions. Context tokens contain locations and response-order features; candidate tokens contain locations, TabPFN predictive means and standard deviations in the online representation space, and a persistent-pool indicator. Separate attention streams encode context and candidates before candidate-to-context attention forms action scores.

The candidate construction, network design, and training objective are shared across experiments. The auxiliary-label parameters in Eq. (8) are $k = 20$, $\alpha = 0.35$, $\epsilon_\tau = 10^{-6}$, and $\zeta = 0.001$; the value, entropy, and auxiliary coefficients in Eq. (10) are $c_V = 0.5$, $c_{\text{ent}} = 0.01$, and $\lambda_{\text{aux}} = 0.2$. PPO uses complete-trajectory returns without discounting.

B EXPERIMENTAL PROTOCOL

This appendix expands the experimental setup in Section 4.1. The practice-function sampler and policy implementation are described in Appendix A; the benchmark task generators and evaluators are separate from that training procedure.

B.1 TASK ARCHIVES AND EVALUATION BUDGETS

Each benchmark provides a fixed archive of $K = 20$ related historical tasks, with $5d$ observed input–response pairs per task. The five synthetic families each have 100 target tasks with identities distinct from those in the corresponding historical archive. BOPTEST and SU2 each have 20 historical and 20 target tasks. Decision variables are mapped to $[0, 1]^d$ within each benchmark. The dimensions are $d = 6$ for Hartmann6, Ackley6, Levy6, and BOPTEST; $d = 10$ for SU2; and $d = 20$ for Ackley20 and Levy20.

At each target task, the methods receive the same initial design of $n_0 = d/2$ scrambled Sobol points and a total budget of $B = 5d$ objective evaluations, including that design. Thus (n_0, B) is $(3, 30)$, $(5, 50)$, or $(10, 100)$ for 6D, 10D, or 20D tasks, respectively. The reported target-task runs use evaluation seed 42. Every target-task objective evaluation, including the initial design, counts toward B; constructing and querying practice functions during training are computational operations. Once an archive is frozen, no method can request additional evaluations of a historical task. The synthetic reference values used for scoring are withheld from all optimizers and are defined in Appendix C.

B.2 SYNTHETIC TASK CONSTRUCTION

Hartmann6, Ackley6, Levy6, Ackley20, and Levy20 denote related-task families, not repeated optimization of a single unmodified benchmark function. All tasks are defined on $[0, 1]^d$. A benchmark generator samples each task's parameters and provides objective responses to the historical logger or the target-task evaluator; neither the parameters nor unqueried responses are available to CAP training. Historical and target tasks have distinct identities, and all 100 sampled targets in each family are retained without selection by method performance.

Task variation. Hartmann6 retains the four basin centers P_i and base precision coefficients A_{ij} of the Hartmann function. For task parameters θ, its objective has the form

$$f_\theta(x) = s_\theta \left[-\sum_{i=1}^{4} \alpha_i^{(\theta)} \exp\left(-\sum_{j=1}^{6} A_{ij} m_i^{(\theta)} c_j^{(\theta)} \big(W_\theta(x)_j - P_{ij}\big)^2 \right) \right]. \tag{14}$$

Here W_θ is a boundary-preserving invertible warp, while $\alpha_i^{(\theta)}$, $m_i^{(\theta)}$, and $c_j^{(\theta)}$ vary basin depth, width, and coordinate anisotropy. The base depths are $(1, 1.2, 3, 3.2)$. Warp shifts, scales, and pairwise coupling angles range over $[-0.0354, 0.0354]$, $[0.9056, 1.0944]$, and $[-11.8°, 11.8°]$, respectively; depth factors range over $[0.5, 1.5]$. Width and anisotropy receive smaller task-specific perturbations. The generator-only factor s_θ keeps the numerical reference minimum on the base Hartmann scale. Neither s_θ nor the task parameters are supplied to an optimizer.

For Ackley and Levy, a task combines a base function, the same type of invertible warp, and a localized smooth perturbation. Let R map unit-cube coordinates to the corresponding base-function domain, coordinatewise. Then

$$f_\theta(x) = f_{\text{base}}(R(z)) + A_\theta \exp\left(-\frac{\|z - z^{\text{opt}}\|^2}{2L^2} \right) \phi_\theta(z), \qquad z = W_\theta(x). \tag{15}$$

Specifically, $R(z) = 10z - 5$ for Ackley and $R(z) = 20z - 10$ for Levy, giving base-function domains $[-5, 5]^d$ and $[-10, 10]^d$, respectively; z^{opt} is the corresponding base optimum in unit-cube coordinates. The localized perturbation ϕ_θ uses 32 cosine features with task-specific coefficients and envelope width $L = 0.18\sqrt{d/6}$. Its amplitude is 0.3 times the family reference scale for Ackley and 0.1 for Levy. Warp shifts, scales, and coupling angles range over $[-0.018, 0.018]$, $[0.952, 1.048]$, and $[-6°, 6°]$, respectively. These variations produce distinct related tasks within a common decision space.

B.3 Engineering Simulation Tasks

BOPTEST evaluates a six-parameter building-control problem using the `multizone_office_simple_air` FMU in BOPTEST v0.9.0. Tasks correspond to different operating windows in the same building and Chicago representative weather year. Each window has a seven-day warm-up followed by three scored days at one-hour control intervals. The six decision variables are occupied heating and cooling setpoints (18–22°C and 22–26°C), unoccupied heating and cooling setpoints (12–18°C and 26–32°C), supply-air temperature (12–18°C), and duct static pressure (100–350 Pa). A shared coordinate mapping enforces heating–cooling deadbands of at least 2°C when occupied and 4°C when unoccupied. The minimized scalar objective is $\mathrm{cost}_{\mathrm{tot}} + \mathrm{tdis}_{\mathrm{tot}} + \mathrm{idis}_{\mathrm{tot}}/494$; it is not normalized by a task-specific optimum.

SU2 evaluates ten-dimensional RAE2822 airfoil designs with SU2 v8.5.0, an SA turbulence model, fixed target lift coefficient, and cold-start RANS evaluations from a shared mesh. The design coordinates control five Hicks–Henne deformations on each of the upper and lower surfaces, each within ± 0.001 chord. Tasks vary flight conditions: Mach number in $[0.79, 0.80]$, Reynolds number in $[6 \times 10^6, 7 \times 10^6]$, and target lift coefficient in $[0.70, 0.74]$. The benchmark minimizes a prespecified scalar objective: feasible simulations use $C_D/(1 + C_D)$ when $C_M < 0.093$ and maximum thickness $t_{\max} > 0.12$; constraint violations receive $1 + v/(1 + v)$, where $v = \max\{0, C_M/0.093 - 1, 1 - t_{\max}/0.12\}$, and classified geometry or numerical failures receive 2. The original scalar objectives, rather than regret to an unknown optimum, are reported for both simulators.

B.4 Baseline Information and Training

The single-task methods EI, PI, UCB, RolloutEI-2, FunBO, and TuRBO use the current target's observations and their native surrogates and search procedures. TAF-ranking, RGPE, ABLR, and FSBO use the fixed historical records through their transfer mechanisms. FSAF, MetaBO, and NAP are trained on the same record-derived, queryable practice-function distribution as CAP, while retaining their own predictors, candidates, policies, and learning objectives; these practice queries do not access historical or target simulators. PFNs4BO uses its released externally pretrained model. CAP uses a separate frozen, externally pretrained TabPFN predictor for online features and learns its candidate policy from the historical archive. All methods retain their native target-task decision procedures, and only actual target-objective evaluations are budget-matched.

C Synthetic Evaluation and Complete Trajectories

For target task i, let $b_{i,t}$ be the lowest objective value observed through evaluation t. Given the task's reference minimum m_i^{ref} and maximum M_i^{ref} for evaluation, normalized simple regret is

$$r_{i,t} = \frac{b_{i,t} - m_i^{\mathrm{ref}}}{M_i^{\mathrm{ref}} - m_i^{\mathrm{ref}}}.$$

The reference values are used only for offline scoring and are not supplied to the optimizers; reference minima obtained by finite search are not claimed to be certified global optima. At each evaluation count, we take the median of $r_{i,t}$ across the 100 target tasks in a family. The horizontal axis includes evaluations used by the initial design, so trajectories begin once that design is complete. Table 1 reports final-budget values; Figures 4–7 give the complete trajectories by method category under the protocol in Appendix B.

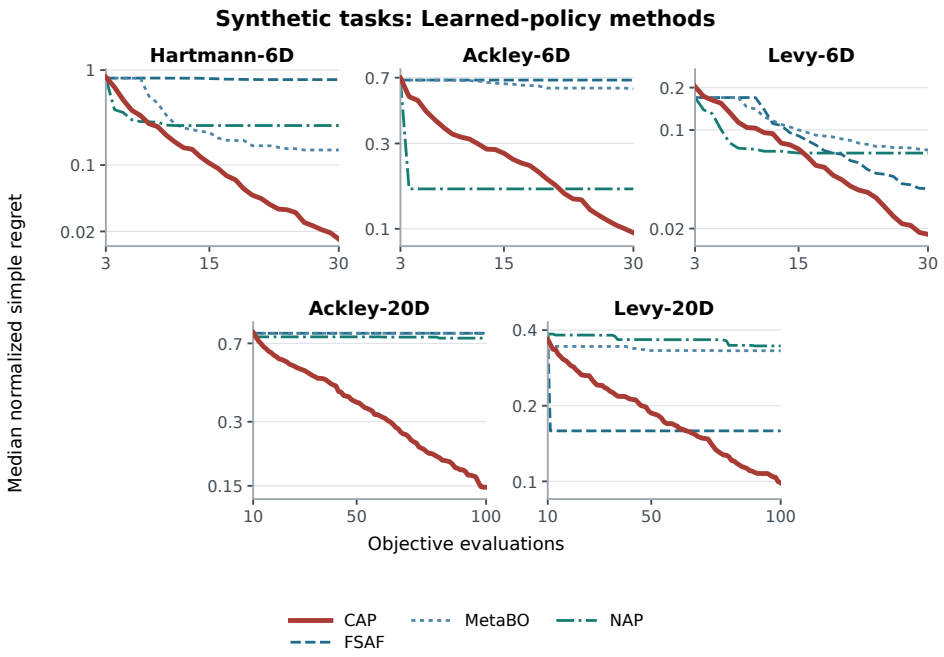

Figure 4: Median normalized simple regret for CAP and the learned query policies FSAF, MetaBO, and NAP across the five synthetic families. Lower is better; vertical scales differ by family.

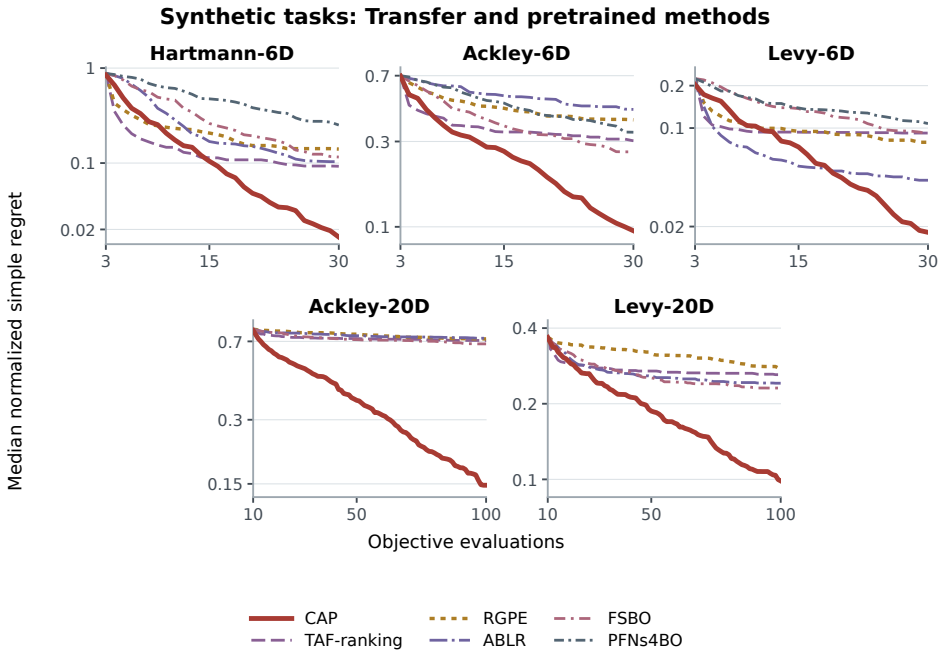

Figure 5: Median normalized simple regret for CAP, TAF-ranking, RGPE, ABLR, FSBO, and PFNs4BO. The released PFNs4BO model supports at most 18 input dimensions, so it has no 20D curves.

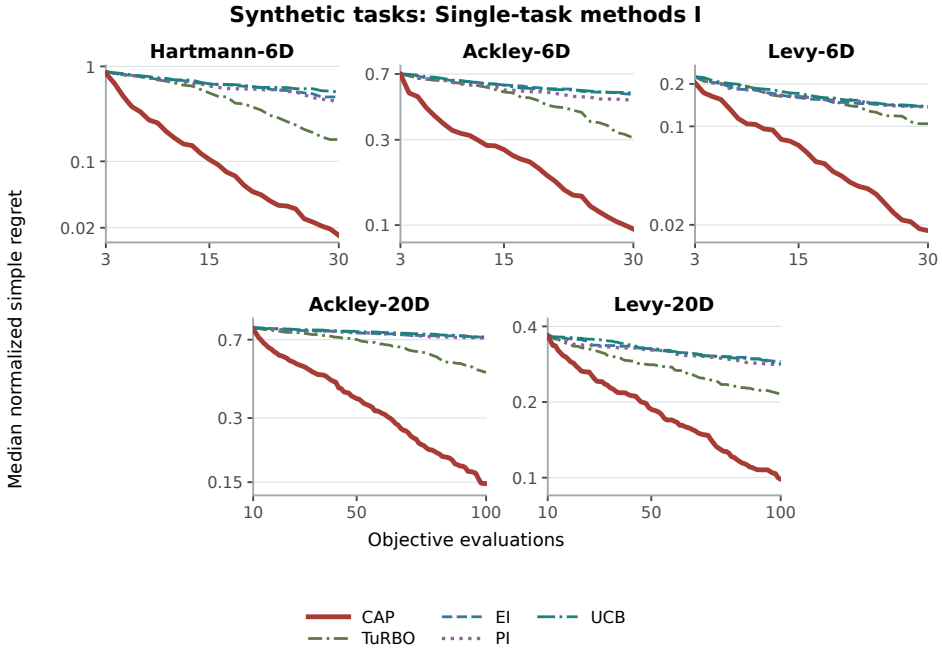

Figure 6: Median normalized simple regret for CAP, TuRBO, EI, PI, and UCB. The evaluation count includes the initial design.

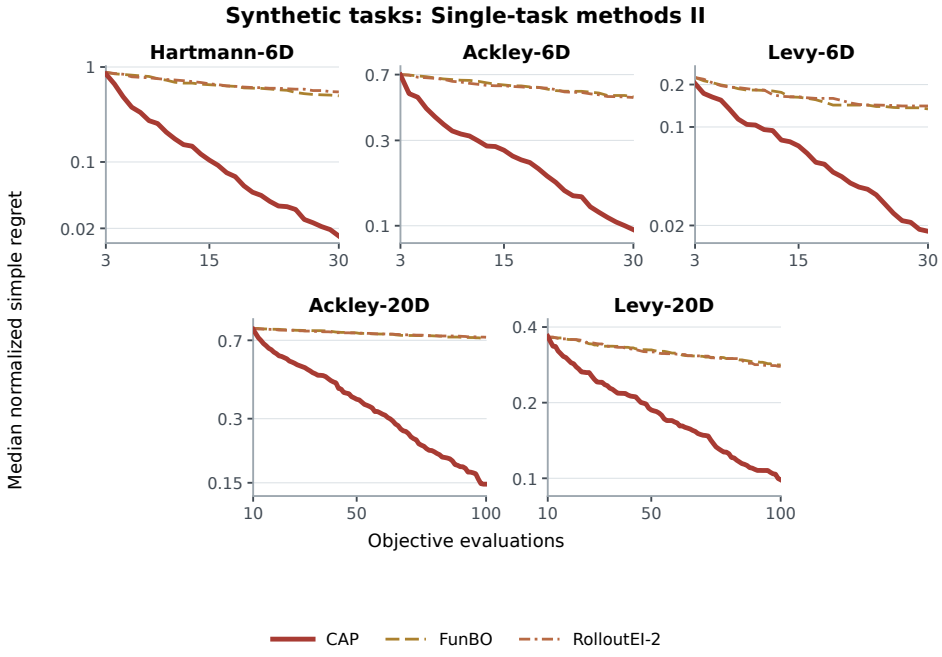

Figure 7: Median normalized simple regret for CAP, FunBO, and RolloutEI-2. Each point is a cross-task median at the indicated evaluation count.

D Complete Engineering Simulation Trajectories

The engineering benchmarks have no known global optimum from which to compute regret, so we use the original objective, as in the main text. For target task i, let $y_{i,s}$ be the objective value at simulator evaluation s and $b_{i,t} = \min_{1 \le s \le t} y_{i,s}$ the best value observed through evaluation t. At each t, the curves report the median of $b_{i,t}$ over the 20 target tasks; lower is better. Both horizontal axes count all simulator evaluations. The BOPTEST curves begin after the three-point initial design. To make subsequent changes easier to discern, the SU2 plots show evaluations 6–50; its five initial evaluations still count toward the total budget. Table 2 gives the final-budget comparison, while Figures 8–11 show every reported method by category.

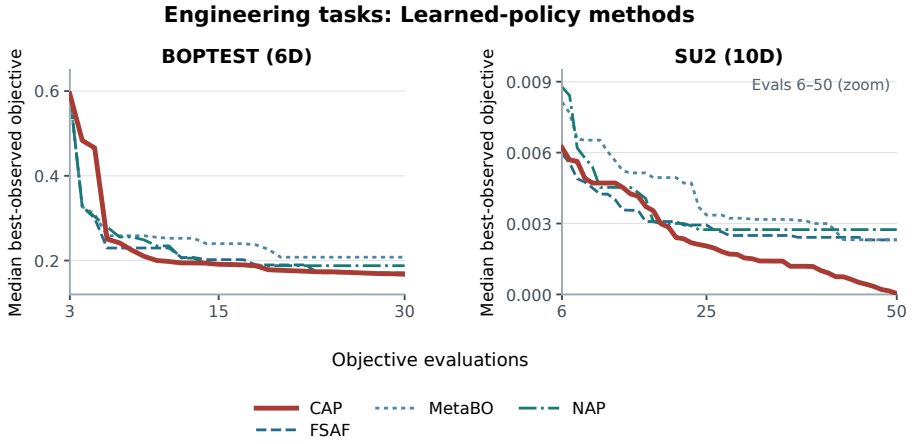

Figure 8: Median best-observed objective for CAP and the learned query policies FSAF, MetaBO, and NAP on BOPTEST and SU2.

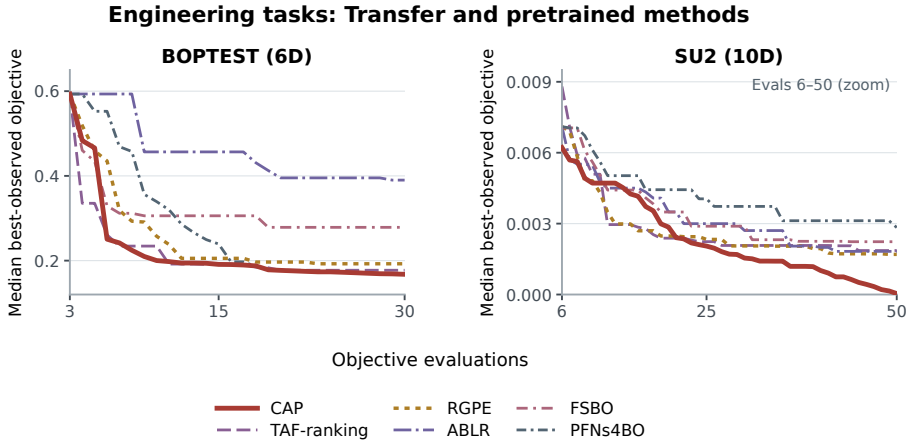

Figure 9: Median best-observed objective for CAP, TAF-ranking, RGPE, ABLR, FSBO, and PFNs4BO on both engineering benchmarks.

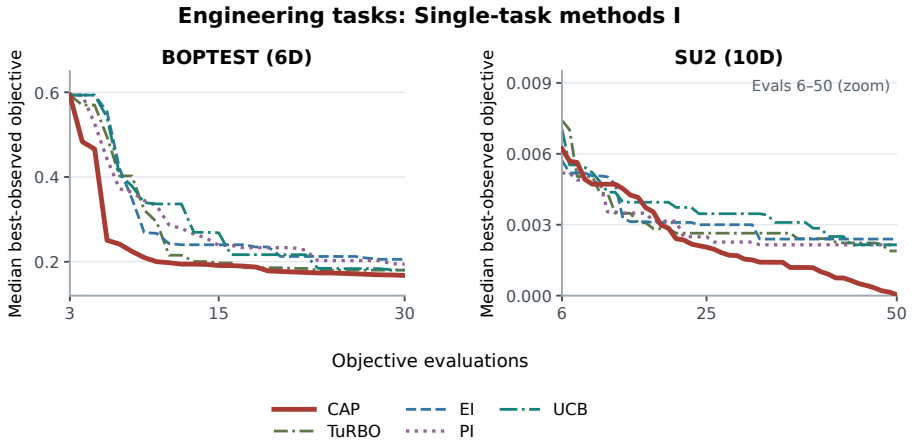

Figure 10: Median best-observed objective for CAP, TuRBO, EI, PI, and UCB on both engineering benchmarks.

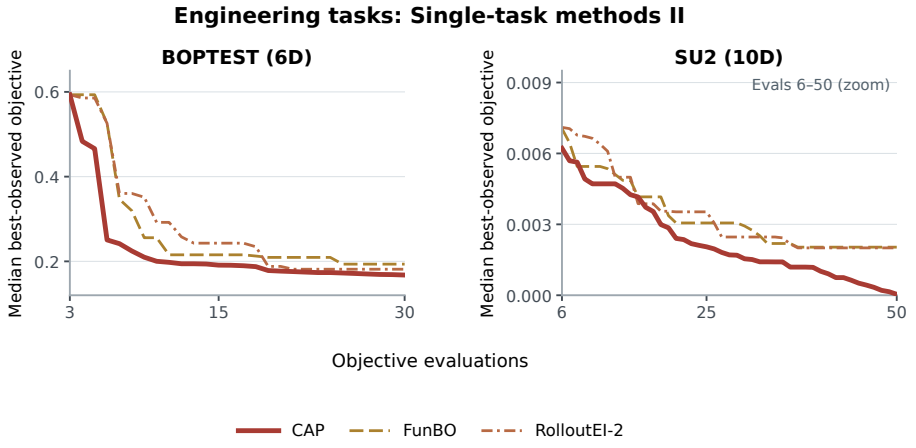

Figure 11: Median best-observed objective for CAP, FunBO, and RolloutEI-2 on both engineering benchmarks.