HW3_20230508_SominLee

1. High-level strategy and objectives of project

1.1. Objectives :

1. **Follow the best offline design.**

   Use the offline log to search for new timetables whose **predicted** exam scores are at least comparable to the best score observed in the data, while controlling risk.

2. **Quantify the quality and risk of the best candidates.**

   For each model, compute the **mean and standard deviation of the top-k candidates** (under the surrogate posterior) so that one can personally judge the trade-off between high expected score and risk.

3. **Learn well on offline data while avoiding over-optimistic OOD behavior.**

   The surrogate must fit the offline log accurately, explore multiple promising regions, and still avoid being disturbed by out-of-distribution (OOD) points where the model is over-confident.

1.2. Surrogate model-based

Two main approaches are mentioned:

1. **Surrogate model-based**: Learn $\hat{f}(x) \approx f(x)$ from D, **optimize the surrogate** instead of the unknown true function.
2. **Generative model-based**: Learn (approximate) inverse $g(y) \approx \text{argmax} f(x)$ and sample good designs for a target score.

The assignment explicitly emphasizes approximating the black-box function from offline data and then optimizing it rather than training a powerful generator. A surrogate can directly inspect quantities like **posterior mean μ(x)**, **uncertainty σ(x)**, and **distance to the offline dataset**, which is crucial for OOD robustness.

Surrogate models integrate naturally with classical optimization tools and allow **trajectory-based analysis** in PCA space: visualize how candidate designs move from random initialization toward high-score regions for each model. Given the relatively small offline dataset, a full generative model is likely harder to regularize for OOD behavior.

1.3. Research question.

> How do different surrogate models (MLP, GP + NeMO, RoMA, COMs) trade off expected score and OOD-robustness when optimizing timetables from a fixed offline log?

Following Steps:

1. **Build a MLP baseline, GP baseline with an ARD RBF kernel.**

   Vanilla Gaussian process serves as a reasonably calibrated Bayesian surrogate over the 5-dimensional design space. MLP is a simple baseline.

2. **Construct three conservative GP-based surrogates over two baselines.**
   - **NeMO**: Penalizes OOD regions using a conservative objective.
   - **COMs**: Strongly conservative model that looks at worst-case GP predictions in a local neighborhood to avoid fragile, spiky optima.
   - **RoMA**: Robustness-aware model that averages predictions under local perturbations and uses the variance of these perturbed predictions as a risk measure.

3. **Compare the models along several axes.**
   - **Validation error, Optimization trajectories in PCA**
   - **Risk–reward characteristics of the final candidates**: analyze the **posterior mean μ**, **uncertainty σ**, and **distance from the offline dataset.**

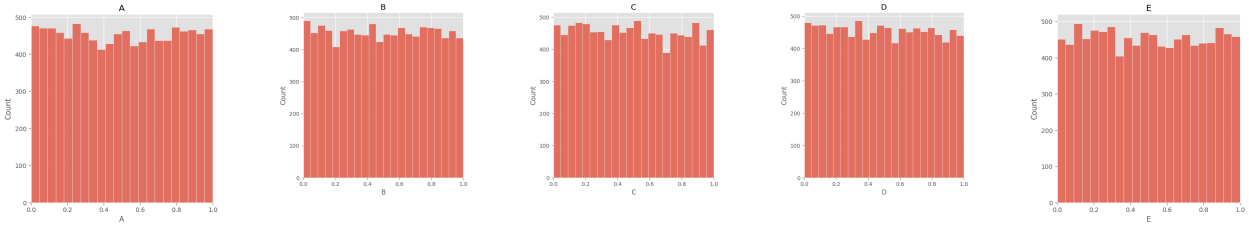
2. Data and Constraint Interpretation

2.1. Problem Definition

README suggests a natural 24-hour budget constraint on the unormalized variables.

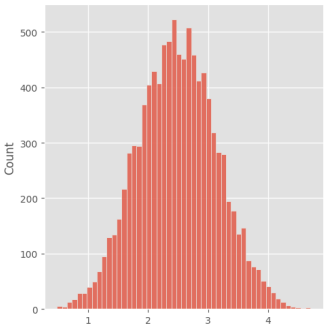
$$A_{hr} + B_{hr} + C_{hr} + D_{hr} + E_{hr} \leq 24.$$

However, the inputs are already **normalized** to the interval [0,1], and uniformly sampled.



Also, the distribution of the sum of these are shown like below

$$min \sum_i x_i \approx 0.49$$
$$max \sum_i x_i \approx 4.70$$
$$E[min \sum_i] x_i \approx 2.49$$



1. **The dataset is scaled**, not in real hours.(scale factor unknown)
2. The story "24h timetable" is a **semantic constraint,** so the constraints stays

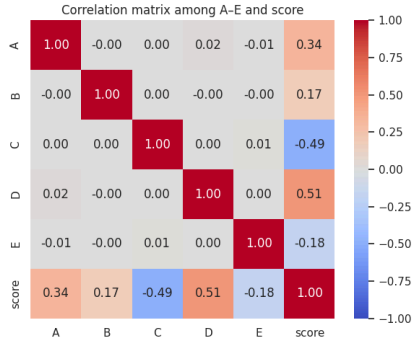
$$x_i \in [0,1] \text{ for i=1,2,3,4,5, } \sum x_i \leq \text{scaled 24 hours(semantic)}$$

2.2. Dataset Structure

For the Correlation matrix for A,B,C,D,E, off-diagonal elements are all **near 0,** so the five activities in the given data were sampled independently.

Correlation of each variable with `score`

- A and D: clearly **positive**
- C (rest): clearly **negative**
- B (examples): mild positive
- E (sleep): mild negative



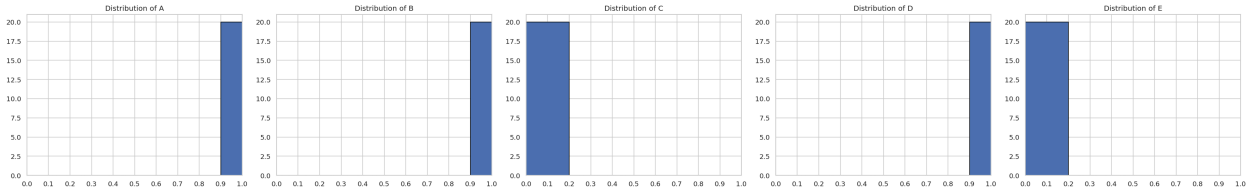
There are infinitely many possibilities but only 9,992 samples, and the **points near the max score is rare.** So the feasible region for the real problem is almost entirely **OOD** with respect to the offline dataset, which motivates using uncertainty and conservatism-based methods (NEMO, COMs, RoMA) rather than naive maximization.

3. Baseline MLP Surrogate and implementation:

3.1. Introduction : what happened before and after normalization

The provided baseline code is an 2-layer MLP on the **raw exam scores** (no normalization). On the test set the RMSE **≈ 1.3 points**, which is a reasonable value.

However, when MLP **optimize timetables by gradient ascent**, almost every trajectory ended in a **corner of the search space** (very extreme A/D, almost zero rest, etc.). These optima was clearly out-of-distribution.

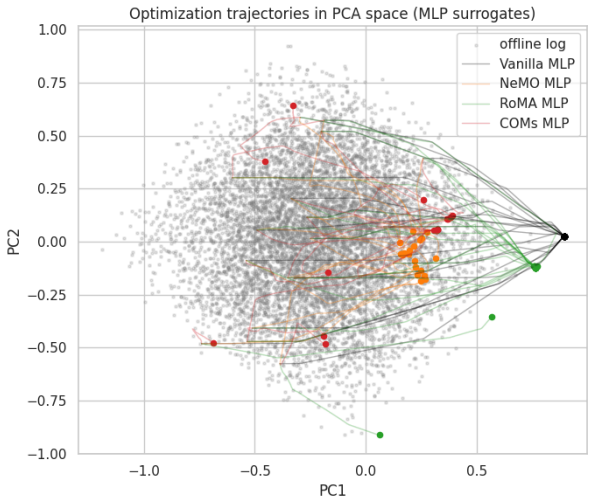


After **standardizing target ;**

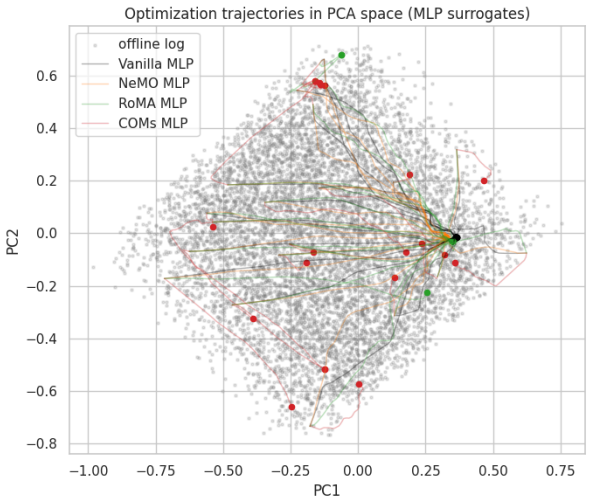
$$y_{norm} = \frac{(y - mean)}{std}, \text{with mean} \approx 70, \text{std} \approx 4$$

The test RMSE on the normalized scale was about **0.44**, which corresponds to roughly **1.8 points** in the original scale – similar quality to the un-normalized model.

In contrast, gradient ascent trajectories **stayed near the data cloud** and converged to reasonable timetables like (0.6, 0.3, ...) instead of extreme corners.



PCA : offline space & raw MLP



PCA : offline space& normalized MLP

As a predictor, both MLPs are similar. As an **optimizer**, the raw-scale MLP is unstable and pushes into OOD corners, while the normalized MLP stays inside the distribution.

Why does label normalization change the optimum?

MLP : f(x; θ) = $w^\top$ h(x) + b, where h(x) is the hidden representation and (w, b) is the last linear layer.

Un-normalized case

- The loss forces f(x) to cover mean ~ 70, so the last-layer weights w tend to become **large**. Outside the training region, f(x) can increase very quickly, and gradient ascent runs along that direction until it hits a **boundary**.

Normalized case

- Gradients of the loss are much smaller in scale. Training prefers solutions with **smaller w**, a smoother function with smaller gradients in x. Outside the data region the function grows more slowly, so gradient ascent tends to stop near high-density regions of the log.

Because the MLP is trained by non-convex optimization (SGD/Adam), un-normalized has large weights and aggressive extrapolation (corner optimum), normalized has smaller weights and smoother extrapolation (in-distribution optimum). This is why the argmax can change, and we should normalize the data before training for conservative OOD.

3.2. Simple Implementation of MLP Surrogate : NeMO, COMs, RoMA like surrogate

We use a single MLP backbone and attach different “GP-inspired” heads.

In the MLP surrogates we replace the GP mean μGP(x) by a neural predictor and then design each head so that it behaves like a neural approximation to μGP(x) and/or σGP(x) and the corresponding GP acquisition.

Shared MLP Backbone

$$
\begin{gathered}
h^{(0)}(x) = x \\
h^{(\ell)}(x) = \phi\Big(W^{(\ell)} h^{(\ell-1)}(x) + b^{(\ell)}\Big), \quad \ell = 1, \ldots, L-1 \\
z(x) = h^{(L-1)}(x; \theta_{\text{back}})
\end{gathered}
$$

NeMO head (mean + variance, GP LCB-like)

In a GP we have a posterior mean mu_GP(x) and variance sigma_GP^2(x), and we often use an LCB

$$LCB_{GP}(x) = mu_{GP}(x) - kappa * sigma_{GP}(x).$$

To mimic this with an MLP, we attach a 2-dimensional NeMO head on top of the backbone feature z(x):

$$\begin{pmatrix} \mu_{\text{norm}}(x) \\ \log v_{\text{norm}}(x) \end{pmatrix} = W_{\text{NeMO}}\, z(x) + b_{\text{NeMO}}$$

$$\sigma_{\text{norm}}(x) = \sqrt{v_{\text{norm}}(x)} = \exp\big(\tfrac{1}{2}\log v_{\text{norm}}(x)\big)$$

The NeMO loss is the Gaussian negative log-likelihood on the normalized scale:

$$L_{\text{NeMO}} = -\sum_i \log N(y_{i,\text{norm}} \mid \mu_{\text{norm}}(x_i), v_{\text{norm}}(x_i))$$

$$J_{\text{NeMO}}(x) = \mu(x) - \kappa\,\sigma(x)$$

COMs head (local worst-case, GP-COM-like)

In GP-based COMs conservative estimate of the local mean is the objective. To imitate a small neighborhood B(x) we define

$$f_{COMs}(x;\theta) \approx y_{norm}$$

During training we sample adversarial directions delta_k around each x_i.

$$\delta_k \in \mathbb{R}^5, \quad \|\delta_k\|_2 = \delta, \quad x_{i,k}^{\text{adv}} = \Pi_{[0,1]^5}(x_i + \delta_k)$$

$$y_{clean,i} = f_{COMs}(x_i;\theta),\ y_{adv,i,k} = f_{COMs}(x_i + \delta_k;\theta),$$

where delta_k have fixed norm delta and are projected back to [0,1]^5.
The COMs loss is

$$\sigma_{\text{COMs-MLP}}(x) := \max\{\varepsilon,\ \mu^{\text{MLP}}(x) - \mu_{\text{adv}}^{\text{COMs}}(x)\}, \quad \varepsilon = 10^{-6}.$$

$$\text{LCB}_{\text{COMs-MLP}}(x) = \mu_{\text{adv}}^{\text{COMs}}(x) - \kappa\ \max\{\varepsilon,\ \mu^{\text{MLP}}(x) - \mu_{\text{adv}}^{\text{COMs}}(x)\}.$$

RoMA head (local smoothness, GP-smoothing-like)

A GP with an RBF kernel encodes smoothness: nearby inputs should have similar outputs. In GP-based RoMA we look at local averages of the GP posterior mean under perturbations.

$$\mu_{\text{norm}}^{\text{RoMA}}(x) = w_{\text{RoMA}}^{\top} z(x) + b_{\text{RoMA}}$$

$$\text{noise}: \xi_i \sim \mathcal{N}(0, \sigma_\xi^2 I), \quad x_{i,\text{noisy}} = \Pi_{[0,1]^5}(x_i + \xi_i)$$

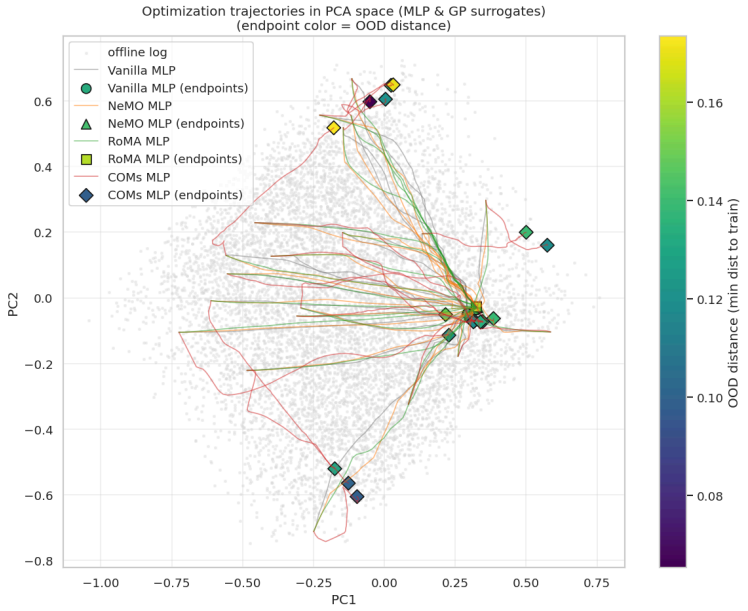
$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N} \big(\ \mu_{\text{norm}}^{\text{RoMA}}(x_i) \ - y_{i,\text{norm}}\ \big)^2$$

$$\tilde{\mu}_{\text{norm}}^{\text{RoMA}}(x_i) := \mu_{\text{norm}}^{\text{RoMA}}(x_i) \quad (\text{smooth\_penalty에서는 gradient X})$$

$$\text{smooth\_penalty} = \frac{1}{N}\sum_{i=1}^{N}\left(\mu_{\text{norm}}^{\text{RoMA}}(x_{i,\text{noisy}}) - \tilde{\mu}_{\text{norm}}^{\text{RoMA}}(x_i)\right)^2$$

$$L_{\text{RoMA}} = \text{MSE} + \beta_{\text{smooth}} \cdot \text{smooth\_penalty}$$

3.3. How NeMO/COMs/RoMA works



MLP surrogate with kappa = 1

- **NeMO** endpoints stay relatively close to the offline cloud.
- **RoMA** trajectories go further, but still mostly follow the shape of the data.
- **COMs** is most “aggressive”: farthest from the training set

3.4. Why do we still need GP surrogates?

Even with normalization, the MLP is still a **deterministic** model. We still introduce a GP layer to obtain calibrated uncertainty for conservative objectives (see Sec. 4.1).

In this sense, the GP models are not meant to replace the MLP-based NeMO/RoMA/COMs surrogates, but to complement them as a probabilistic “judge”. The MLP surrogates are used to generate strong candidate timetables via gradient-based search, while the GP posterior mean and variance provide a calibrated, common reference for constructing conservative objectives such as LCB, risk–reward plots, and efficient frontiers.

4. Building the Gaussian Process(GP) regression

4.1. Motivation for GP regression

4.1.1 Why Gaussian Process as the base surrogate?

- GP regression is a **nonparametric Bayesian regression** technique that places a Gaussian process prior over functions f(·)
- It gives a **posterior mean** μ(x | D) and **posterior variance** σ2(x | D) for any input x.

$$\mu(x|D) = k(x,X)^T(K+\sigma_\varepsilon^2 I)^{-1}y$$
$$\sigma^2(x|D) = k(x,x) - k(x,X)^T(K+\sigma_\varepsilon^2 I)^{-1}k(x,X)$$

GP is effective for the base surrogate for the below reasons :

1. **Low-dimensional input (D=5)** and **moderate dataset size (≈10k)**
2. **Care about uncertainty**, not just a point prediction.
    - In offline model-based optimization, naive maximization of a point-estimate model tends to exploit **overconfident errors** on OOD inputs.
    - Sampler can provide the std, mean of top-k data in order to make a decision by oneself.
3. **Interpretability**
    - With an ARD RBF kernel, get an interpretable lengthscale per feature: how sensitive the exam score is to each activity.
4. **GP fits better**

```
Vanilla_MLP test_RMSE= 0.408  R2= 0.991
GP          test_RMSE= 0.393, R2= 0.992
```

4.1.2 GP modeling details

1. **Preprocessing**
    - Normalize inputs to [0,1] (already true)/Standardize the score y
2. **Kernel choice**
    - Use **ARD RBF kernel**

$$k(x,x') = \sigma_0^2 * exp(-0.5 * \Sigma_d((x_d - x'_d)^2/\lambda_d^2))$$

    - Using an ARD RBF kernel, the learned length-scales for A-E show that the exam score is most sensitive to B (extra examples), but similar around all A-E. This pattern is different with the simple Pearson correlations in Figure 3, because **reflects the fully nonlinear GP surrogate rather than just linear dependence.**

```
0.798**2 * RBF(length_scale=[0.565, 0.566, 0.596, 0.524, 0.563])
```

3. **Hyperparameter tunning (in kappa, 5.1.)**

   Omitted in this paper for other hyperparameter tunnings.

4. **Early Stopping**(also done in MLP surrogates)

5. **Posterior prediction**

   - For any new timetable x, GP returns μ(x): predicted score, σ(x): uncertainty.

4.2. Conservative surrogates: NeMO, COMs, and RoMA

NeMO, RoMA, and COMs build conservative search objectives from the GP posterior, typically of the form μ(x)−κσ(x).

$$\text{Conservative acquisition form: } f_{hat}(x) = \mu(x) - \kappa * \sigma(x)$$
$$\text{All three conservative models can be written as:}$$
$$f_{hat_{model}}(x) = \mu_{model}(x) - \kappa * \sigma_{model}(x)$$

1. **NeMO (Normalized Maximum Likelihood / CNML)**

   - Adjusts the predictive distribution such that points with little support in the training data receive **high normalized uncertainty**.

$$\text{normalization penalty} : \rho(x) = \frac{p_{data}(x)}{p_{prior}(x)}$$
$$\mu_{NeMO}(x) = \mu_{GP}(x) - \kappa * \sigma_{GP}(x) * g(\rho(x))$$

2. **COMs (Conservative Objective Models)**

   - Uses adversarially generated neighbors of the log points to train a **lower-bounding** proxy.

$$\text{Adversarial neighborhood: } B_\delta(x) = x' : ||x' - x||_2 \leq \delta$$
$$\mu_{COMs}(x) = min_{x' \in B_\delta(x)} \mu_{GP}(x')$$
$$\sigma_{COMs}(x) = \mu_{GP}(x) - \mu_{COMs}(x)$$

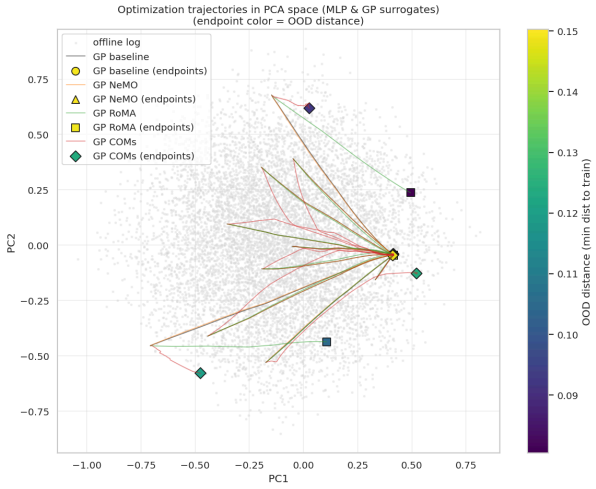
3. **RoMA (Robust Model Adaptation)**

   - Adds local smoothing around the current candidate during gradient ascent by injecting input noise and adversarial weight perturbations.

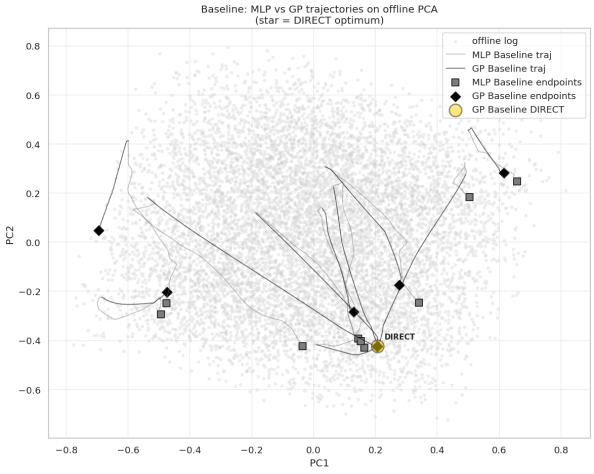
$$\mu_{RoMA}(x) = E_{\xi\ N(0,\sigma_\xi^2 I)}[\mu_{GP}(x+\xi)]$$
$$\sigma_{RoMA}(x) = sqrt(Var_{\xi\ N(0,\sigma_\xi^2 I)}[\mu_{GP}(x+\xi)])$$

Together, these methods complement the GP baseline by explicitly encoding where the data is located and how much we trust the model outside that region.

4.3. How NeMO/COMs/RoMA works



GP surrogates with Kappa = 1



Compare DIRECT, GP, MLP COMs

Because the GP surrogate is not differentiable, I approximated a gradient-like update rule to visualize the optimization trajectory. Under this heuristic, NeMO tends to move directly toward the global optimum, whereas COMs explores the space more aggressively. However, that these trajectories are not driven by true gradients, as they are not defined for this surrogate model.

To address this limitation, I additionally used the **DIRECT global optimization algorithm** on each surrogate. DIRECT systematically explores the entire [0,1]^5 design space by recursively partitioning it into hyper-rectangles, which gives a diverse pool of high-scoring candidates that better approximate the global search landscape.

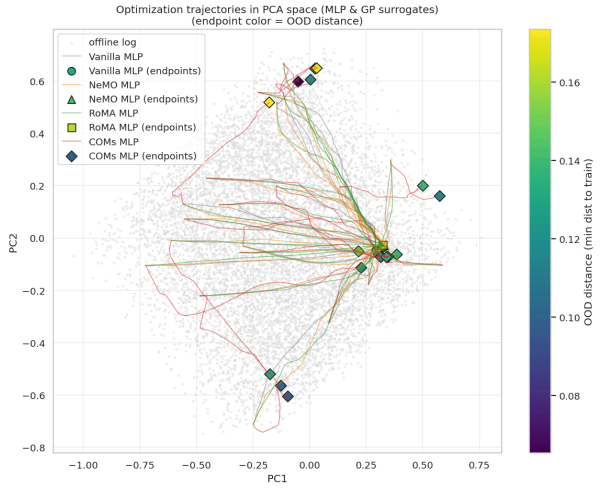
5. Common points of GP- and MLP-based conservative surrogates

Two different NeMO, RoMA, and COMs implemented :

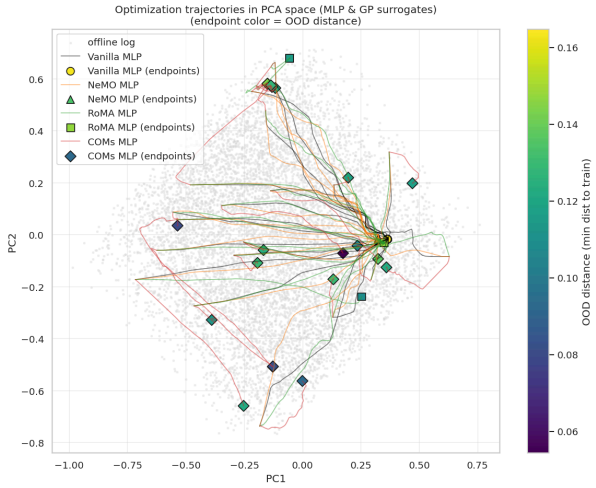
- as **MLP-based surrogates** that add NeMO/COMs/RoMA-style losses
- as **GP-based surrogates** that transform the posterior mean and variance of a Gaussian process.

5.1. Local vs Global search effect by conservatism parameter κ

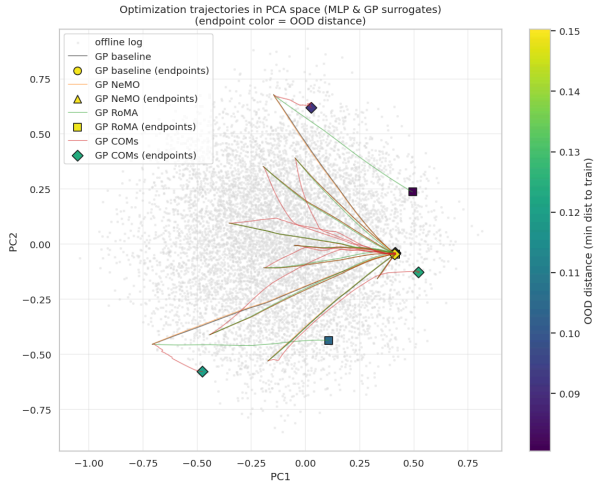
Conceptually, κ controls the **trade-off between exploring many local opportunities and aggressively chasing a single high-score basin**:



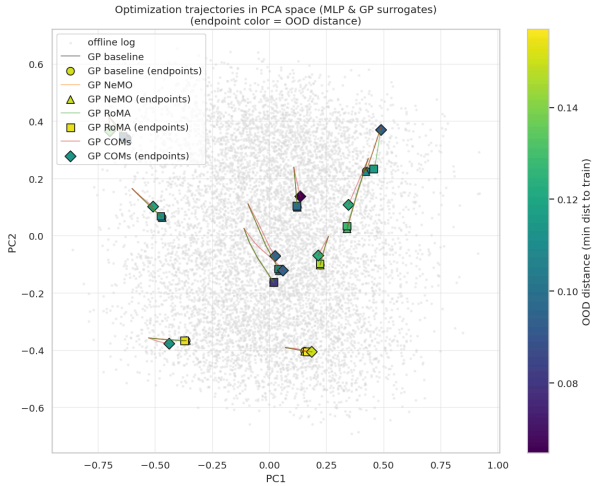
MLP surrogates with Kappa = 1



MLP surrogates with Kappa = 3



GP surrogates with Kappa = 1



GP surrogates with Kappa = 3

- **κ↓ :** Likely acts as Vanilla model, tends to converge to global optima.
- **κ↑ :** Because the predictive uncertainty σ(x) grows rapidly at the offline data manifold, these objectives strongly penalize unsupported regions and prevent the emergence of a single dominant OOD peak. **The resulting surrogate landscape contains several moderate local maxima around the data cloud.**

From a gradient perspective, the difference between conservative and non-conservative surrogates becomes pronounced as κ increases. All models effectively optimize an acquisition of the form

$$J(x) = \mu(x) - \kappa\sigma(x), \quad \nabla_x J(x) = \nabla_x \mu(x) - \kappa \nabla_x \sigma(x)$$

Conservative surrogates such as NeMO, RoMA, and COMs are trained so that sigma(x) is large specifically in unsupported or unstable regions, so it actively pushes search away from these areas when κ is large. Vanilla-like surrogates lack this alignment, so even at high κ their updates are still mainly driven by mu(x), which explains why they more often drift toward spurious high-mean, high-uncertainty regions in the experiments.

5.2. Local vs Global search between models

Across both implementations, several qualitative patterns are consistent:

- **NeMOs tends to be the safest.** Its candidates have the smallest OOD distances, but its predicted scores are slightly lower.
- **COMs tends to explore more aggressively.** COMs often finds timetables with higher predicted mean scores while still staying inside the main data cloud.
- **RoMA in between.**

These patterns can be understood through the shared conservative objective J(x) = mu(x) - kappa sigma(x).

- NeMO learns mu(x) and sigma(x) so that uncertainty is inflated in unsupported regions, which makes the term -kappa sigma(x) strongly repel the search from the data boundary, slightly under-exploratory behavior.
- COMs explicitly pushes down locations where adversarial neighbors have larger predicted mean, so it is willing to move toward high-mu(x) regions, leading to more aggressive but still data-aware exploration.
- RoMA regularizes local smoothness and discourages sharp changes under small perturbations, which naturally places it between NeMO and COMs in terms of both OOD distance and achieved scores.

There is no single universally best surrogate; the most appropriate choice **depends on the decision-maker's risk tolerance.**

6. Decision-analytic perspective: how to choose a timetable from the candidates

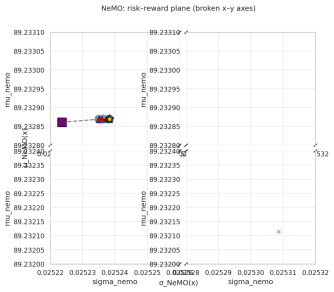
Given the uncertainty of the surrogates and the risk of OOD extrapolation, the crucial question is **how to actually pick a final timetable** from all candidates proposed by NeMO, RoMA, and COMs.

Step 1: Compute risk-aware summaries for each candidate.

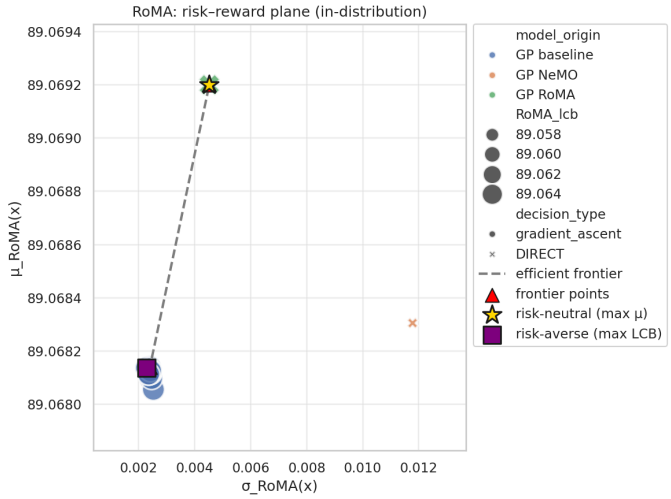
For each candidate timetable x, compute the risk-aware summaries **(μm(x), σm(x), LCBm(x), d(x))** under each judge m ∈ {NeMO, RoMA, COMs}, as defined in Sec. 4.3, Also calculate the **distance d(x) to the offline log.**

Step 2: In-distribution filtering and Pareto frontier.

- Discard candidates : OOD distance d(x)>0.16, LCBm(x)≤best offline score-3
  - in order to keep timetables that are reasonably well supported by data, and guarntees high score
- plot all candidates in the (σm(x),μm(x)) plane for each judge, and compute the **efficient frontier.**



Frontier : NeMO judge



Frontier : RoMA judge

(For COMs, one point dominated all LCBm(x) and μm(x).)

Step 3: Risk-neutral vs risk-averse choices.

For each model m, select two representative timetables:

- a **risk-neutral** candidate: the point on (or near) the efficient frontier with the highest mean μm(x)
- a **risk-averse** candidate: the point with the highest LCBm(x)

If no candidate has LCBm(x) exceeding the best offline score (≈ 89.83), simply select the in-distribution timetable with the largest μm(x) and the largest LCBm(x)

Step 4: Interpreting and comparing the final timetables.

For example, according to the dataset, RoMA suggested two models such as below, where **Risk neutral choice suggests more reading books and less rest & discussion with friends -** which lowers LCB but expects higher mean.

	Reading book	Solve examples	Rest	Discuss with friends	Sleep
Risk - neutral					
Risk - averse					

This connects the technical modeling to a clear decision rule:

> Instead of blindly taking the surrogate's argmax, treat each timetable as a risky option with uncertain payoff and choose those that look good and are well supported by data.

7. Limitations and future directions

Finally, explicitly acknowledging limitations shows depth:

- **Normalization ambiguity** – because we do not know the exact mapping from normalized variables to hours, all results are about *relative* allocations, not about absolute time in hours.
- **Surrogate mis-calibration near the optimum** – all surrogates, including the GP, underestimate the top scores; there is no guarantee that any candidate truly beats the best log point.
- Subsampled 2000 points for GP training(since it takes long time), can try sparse GPs or deep kernel learning,