

Algorithmic Flow: Exploration Strategy (explore)

Original RL vs. two independently evolved CEM-based strategies for gene panel sampling

Original RL

ε -greedy + Gaussian noise

1. Encode state:

$S_t = \text{ENCODER}(\text{stats}(X_{\text{panel}}))$
Train autoencoder for 5 epochs
64-dim latent from 16 global stats

2. Get per-gene scores:

$\text{logits} = \text{ACTOR}(S_t)$
 $p_g = \sigma(\text{logit}_g) \forall g$
Batched actor: $O(G)$ parameters
per gene — wasteful for 2000 genes

3. Sample panel:

With prob ε : add $\mathcal{N}(0, 0.2)$ noise,
target $\sim \mathcal{N}(K_{\text{tgt}}, 100)$
Else: use raw probs, target = K_{tgt}
Sort by (noisy) prob, take top- K
1 panel per step; no diversity control;
noise can produce degenerate panels

4. Evaluate & store:

$r_t = \text{REWARD_PANEL}(\text{genes})$
 $S_{t+1} = \text{ENCODE}(\text{genes}_{t+1})$
Push $(S_t, \mathbf{a}, r_t, S_{t+1})$ to replay buffer
Each call = full PCA+Leiden+ARI;
no caching of repeated panels

5. Repeat for N_{explore} steps;
update `current_subset` each step

$O(N_{\text{explore}})$ expensive evals;
1 panel/step; no proxy screening;
 ε -greedy: blind uniform noise

Explore Best CEM + linear surrogate UCB

1. Build candidate pool (cheap):

Generate $M_{\text{raw}} = 6 \times M$ candidates via:

- Gumbel-TopK on ℓ_g/T (global)
- Local mutation : swap out low- p genes,
swap in high-logit genes via Gumbel-top1
- Prior-preserving : keep 30% prior genes

3 strategies ensure diversity

2. Novelty filter:

$\text{Jaccard}(c, \text{ref}) \geq \theta_{\text{accept}}? \rightarrow \text{reject}$
 θ adaptive: low spread $\rightarrow 0.90$,
high spread $\rightarrow 0.98$
Prevents search collapse to clones

3. Rank by surrogate UCB:

Linear ridge: $\mu = \mathbf{w}^\top \mathbf{x}$, $\sigma = \sqrt{\sum 1/A_{ii}}$
Acquisition = $\mu + \kappa\sigma$
 $-\lambda_{\text{ov}} \cdot \max_j \text{Jaccard}(c, \text{ref}_j)$
No neural net; $O(K)$ per candidate

4. Boltzmann select & evaluate:

Softmax sample M from pool
(adaptive τ : low spread \rightarrow flatter)
Evaluate `REWARD_PANEL`;
memoize by sorted-index tuple
Reduces expensive evals $\sim 80\%$

5. Adaptive temperature kick:

spread $< 0.05 \rightarrow T \times 1.5$
spread $> 0.15 \rightarrow T \times 0.9$
Escapes plateaus dynamically

Optimize Best CEM + neural surrogate UCB

1. Sample via Gumbel-TopK:

$s_g = \ell_g/\tau + \text{Gumbel}(0, 1)$
Take top- K indices as panel
 $\tau = \tau_{\text{base}} \cdot (1 + 1.5\varepsilon)$
 τ_{base} : anneal $1.5 \rightarrow 0.4$ ($\times 0.98/\text{epoch}$)
Plackett–Luce without-replacement

2. Generate large pool:

$M_{\text{pool}} = 6 \times M$ candidates
from Gumbel-TopK at current τ
Single sampling strategy (no mutation)

3. Neural surrogate screening:

Embedding(g) \rightarrow mean-pool \rightarrow MLP
MC Dropout ($T=8$ fwd passes):
 $\mu, \sigma \leftarrow \text{mean}(\hat{r}), \text{std}(\hat{r})$
UCB = $\mu + \lambda\sigma$
Richer model; higher overhead

4. Racing: top fraction evaluated:

Rank by UCB (or PCA proxy);
evaluate top 35% with `reward_panel`
Memoize by sorted-index key
PCA centroid-separation fallback
before surrogate warmup (128 samples)

5. Elitism:

Always evaluate incumbent + best
Update `current_subset` to epoch-best
Guarantees monotonic tracking

No novelty filter;
no adaptive temperature kick;
relies on τ annealing only

Algorithmic Flow: Distribution Update (optimize)

How each variant updates the gene selection distribution after exploration

Original RL Actor-Critic + TD(0)

1. Sample from replay buffer:
 $(S_t, \mathbf{a}, r_t, S_{t+1}) \sim \mathcal{B}$
Minibatch of 64 transitions
Off-policy: transitions from different epochs

2. Critic update (TD(0)):
 $V_t = \text{CRITIC}(S_t)$
 $V_{t+1} = \text{CRITIC}(S_{t+1})$ (no grad)
target = $r_t + \gamma \cdot V_{t+1}$
 $\mathcal{L}_{\text{critic}} = \text{MSE}(V_t, \text{target})$
Bootstrapping off critic's own estimate;
unstable when reward is panel-level
and state transitions are non-Markovian

3. Actor update (policy gradient):
logits = $\text{ACTOR}(S_t)$
 $\log \pi = a_g \log \sigma(\ell) + (1 - a_g) \log \sigma(-\ell)$
 $\mathcal{L}_\pi = -(\log \pi \cdot A_t). \text{mean}$
 $A_t = \text{target} - V_t$ (advantage)
Single-step advantage; high variance;
per-gene $a_g \in \{0, 1\}$ ignores panel-level
combinatorial structure

4. Entropy regularization:
 $H = -(p \log p + (1-p) \log(1-p))$
 $\mathcal{L}_{\text{ent}} = -\beta_{\text{ent}} \cdot H. \text{mean}$
 $\beta_{\text{ent}} = 0.01$ (fixed)
Fixed coefficient; no scheduling

5. $\varepsilon \leftarrow \max(0.10, \varepsilon \times 0.97)$

Policy gradient on per-gene Bernoullis;
critic estimates state-value for a
non-sequential combinatorial problem;
objective–architecture mismatch

Explore Best Logit-space CEM + rank weights

1. Select elites:
Sort population by reward descending
 $E = \lceil \rho \cdot M \rceil$ elites (ρ adaptive)
Low spread $\rightarrow \rho \times 1.25$
High spread $\rightarrow \rho \times 0.9$
Wider elite pool when stagnant

2. Rank-based elite weights:
 $w_i = \exp(-\text{rank}_i / \tau_r)$; normalize
Robust to reward scale; top-ranked
elites contribute more regardless
of absolute reward difference

3. Weighted gene frequencies:
 $\text{freq}_g = \sum_i w_i \cdot \mathbb{I}[g \in \text{panel}_i]$
Vectorized via `np.bincount`
Clip $\geq p_{\min}$; normalize to simplex
 $\hat{\ell}_g = \log(\text{freq}_g)$

4. Logit-space smoothing:
 $\ell_g \leftarrow (1 - \alpha)\ell_g + \alpha\hat{\ell}_g$
Repeat for N_{opt} steps
 α adaptive: stronger when best improves
EMA in log-space prevents simplex
corner collapse; multiple passes
for smoother convergence

5. Recover probabilities:
 $p_g = \text{softmax}(\ell_g)$; clip $\geq p_{\min}$
 $\ell_g \leftarrow \log(p_g)$ (re-sync)

No neural networks in update loop;
no critic; no TD bootstrapping;
pure combinatorial optimization

Optimize Best KL-regularized CEM (trust region)

1. Sample from replay buffer:
 $(S_t, \mathbf{a}, r_t, \dots) \sim \mathcal{B}$; minibatch 64
Standardize: $r \leftarrow (r - \bar{r}) / (\sigma_r + \varepsilon)$

2. Select elites + softmax weights:
 $E = \lceil \rho \cdot |\text{batch}| \rceil$ elites
 $w_i = \text{softmax}(r_i / \tau)$; $\tau = 0.75$
Magnitude-based: sensitive to reward scale

3. Elite empirical marginals:
 $m_g = \sum_i w_i \cdot a_{ig}$
Anti-collapse: $m \leftarrow (1 - \lambda)m + \lambda p_0$
 $p_0 = K/G$; λ decays with ε
Prevents distribution degeneration

4. Trust-region update:
 $p_{\text{old}} = \sigma(\ell_g)$ (current logits)
 $p_{\text{new}} = (1 - \eta)p_{\text{old}} + \eta m$
Check $\text{KL}_{\text{Bern}}(p_{\text{old}} \| p_{\text{new}}) \leq \delta_{\text{KL}}$
If violated: $\eta \leftarrow \eta \times 0.5$ (up to $20\times$)
TRPO-style: guarantees stable updates;
no overshooting in probability space

5. Commit:
 $p_{\text{new}} = \text{clamp}(p_{\text{new}}, \varepsilon, 1 - \varepsilon)$
 $\ell_g \leftarrow \text{logit}(p_{\text{new}})$
 $\tau_{\text{sample}} \leftarrow \max(\tau_{\text{end}}, \tau \times 0.98)$
Temperature monotonically decreases

KL constraint ensures safe updates;
but monotonic τ decay limits ability
to escape local optima once converged

Algorithmic Flow: Reward Function (reward_panel)

How each variant scores a candidate gene panel

Original RL

ARI + size penalty

1. Clustering:
PCA → neighbors → Leiden
Standard scanpy / rapids pipeline

2. Single metric:
ARI = ADJRANDSCORE(true, clusters)
Only ARI; ignores NMI & silhouette

3. Size penalty:
$$s = \begin{cases} 1 & K \leq K_{\text{tgt}} \\ \left(1 - \frac{K - K_{\text{tgt}}}{K_{\text{max}} - K_{\text{tgt}}}\right)^\beta & K_{\text{tgt}} < K \leq K_{\text{max}} \\ 0 & K > K_{\text{max}} \end{cases}$$

4. Final reward:
 $r = \alpha \cdot \text{ARI} + (1 - \alpha) \cdot s$
 $\alpha = 0.8$ (fixed)
Linear combination; equal scaling;
no NMI/SI signal; misses clustering
quality dimensions

Sparse reward signal;
ARI-only misses complementary
quality information from NMI & SI

Explore Best

Multi-metric + nonlinear shaping

1. Clustering: same pipeline +
deterministic Leiden (random_state=0)

2. Multi-metric quality:
ARI, NMI ← sklearn
SI ← silhouette on PCA (subsampled)
 $\text{SI}_s = \tanh(\text{SI})$
 $q = 0.5 \text{ ARI} + 0.3 \text{ NMI} + 0.2 \text{ SI}_s$
Captures complementary quality axes

3. Nonlinear shaping:
 $q' = (\text{clamp}(q, 0, 1))^{1.5}$
Amplifies difference between
good and mediocre panels

4. Missing-gene penalty:
If $K < K_{\text{tgt}}$:
 $\text{pen} = c_{\text{miss}} \cdot (K_{\text{tgt}} - K) / K_{\text{tgt}}$
 $c_{\text{miss}} = 0.05$
Discourages under-sized panels

5. Final reward:
 $r = q' - \text{pen}$
Decoupled from size term s ;
size handled by Lagrangian in trainer

Optimize Best

Multi-metric + power shaping

1. Clustering: same pipeline
Deterministic Leiden for CPU path

2. Multi-metric quality:
ARI, NMI via sklearn
SI: silhouette → $(r + 1) / 2 \in [0, 1]$
 $q_g = w_{\text{ARI}} \cdot f(\text{ARI}) + w_{\text{NMI}} \cdot f(\text{NMI})$
 $+ w_{\text{SI}} \cdot f(\text{SI})$
weights = (0.45, 0.45, 0.10)
More ARI/NMI weight; less SI

3. Per-metric power shaping:
 $f(x) = (\text{clamp}(x, 0, 1))^{2.0}$
Applied to each metric individually
Stronger shaping (power 2.0 vs 1.5);
good metrics amplified more

4. Size penalty:
Same s formula as original
 $\text{pen} = \lambda_s \cdot (1 - s)$; $\lambda_s = 0.25$
Soft Lagrangian rather than hard mix

5. Final reward:
 $r = q_g - \lambda_s \cdot (1 - s)$
Quality dominates; size penalty
only activates above K_{tgt}

Detailed Pseudocode Comparison

Algorithmic differences in exploration (EXPLORE) and optimization (OPTIMIZE)

1. Exploration Strategy (explore)

Algorithm 1a: Original RL — ε -greedy per-gene sampling

Input: Current panel G_t , Actor network, replay buffer \mathcal{B} , ε

Output: Updated panel G_{t+1} , rewards list

```
1: for step = 1, ...,  $N_{\text{explore}}$  do
2:    $S_t \leftarrow \text{ENCODESTATE}(G_t)$  ▷ 16 stats → 64-dim latent
3:   logits  $\leftarrow \text{ACTOR}(S_t)$ ;  $p_g \leftarrow \sigma(\text{logit}_g) \forall g$ 
4:   if rand() <  $\varepsilon$  then ▷ Blind noise exploration
5:      $K \leftarrow \text{clip}(\mathcal{N}(K_{\text{tgt}}, 100), 200, K_{\text{max}})$ 
6:      $\tilde{p}_g \leftarrow p_g + \mathcal{N}(0, 0.2) \forall g$ 
7:   else
8:      $K \leftarrow K_{\text{tgt}}$ ;  $\tilde{p}_g \leftarrow p_g$ 
9:   end if
10:   $G_{t+1} \leftarrow \text{top-}K \text{ genes by } \tilde{p}$ 
11:   $r_t \leftarrow \text{REWARD\_PANEL}(G_{t+1})$  ▷ Expensive: PCA + Leiden + ARI
12:   $\mathcal{B}.\text{push}(S_t, \mathbf{a}, r_t, S_{t+1})$ 
13: end for
```

Algorithm 1b: Explore Best — Two-stage CEM with linear surrogate UCB

Input: Logits ℓ , temperature T , eval budget M , elite archive, best panel G^*

Output: Updated ℓ , elite archive, rewards list

```
1: Stage A: Generate candidate pool ( $M_{\text{raw}} = 6M$ )
2: for  $j = 1, \dots, M_{\text{raw}}$  do
3:   Strategy  $\leftarrow \{\text{global, local-mutation, prior-preserving}\}$  by fraction
4:    $C_j \leftarrow \text{GUMBELTOPK}(\ell/T, K)$  or  $\text{MUTATE}(\text{base, max\_swaps})$  or  $\text{PRIORSAMPLE}$ 
5:   If  $\text{max}_{\text{ref}} \text{Jaccard}(C_j, \text{ref}) \geq \theta_{\text{accept}}$  reject ▷ Novelty filter
6: end for
7: Stage B: Rank & select for expensive evaluation
8:  $\text{acq}_j \leftarrow \mathbf{w}^\top \mathbf{x}_j + \kappa \sqrt{\sum 1/A_{ii,j}} - \lambda_{\text{ov}} \cdot \text{Jaccard}(C_j, \text{refs})$  ▷ Surrogate UCB
9: Eval set  $\leftarrow \text{BoltzmannSample}(\text{acq}, M, \tau_{\text{proxy}}) \cup \{\text{seeds, elites}\}$ 
10: for each  $C \in \text{Eval set}$  do
11:    $r \leftarrow \text{REWARD\_PANEL\_MEMOIZED}(C)$  ▷ Sorted-index cache
12:   Update best, elite archive, reward stats
13: end for
14: Adaptive  $T$ : spread < 0.05  $\rightarrow T \times 1.5$ ; spread > 0.15  $\rightarrow T \times 0.9$ 
```

Algorithm 1c: Optimize Best — Gumbel-TopK + neural surrogate racing

Input: Panel logits ℓ , temperature τ , surrogate model (E, h) , buffer \mathcal{B}

Output: Updated buffer, best panel, rewards list

```
1:  $\tau_{\text{eff}} \leftarrow \tau \cdot (1 + 1.5\varepsilon)$  ▷ More exploratory when  $\varepsilon$  high
2: Generate pool of  $6M$  candidates via  $\text{GUMBELTOPK}(\ell/\tau_{\text{eff}}, K)$ 
3: Surrogate screening: Train surrogate on buffer (adaptive schedule)
4:  $\text{UCB}_j \leftarrow \mu_j + \lambda \sigma_j$  via MC Dropout ( $T=8$  fwd passes on Embed  $\rightarrow$  MLP)
5: If no surrogate:  $\text{score}_j \leftarrow \text{PCA centroid-separation proxy}$ 
6: Evaluate top 35% by score with full  $\text{REWARD\_PANEL}$ 
7: Always evaluate incumbent + best-so-far ▷ Elitism
8: Push transitions to  $\mathcal{B}$ ; update current_subset to epoch-best
```

2. Distribution Update (optimize)

Algorithm 2a: Original RL — Actor-Critic policy gradient + TD(0)

Input: Actor, Critic networks; replay buffer \mathcal{B} ; $\gamma = 0.99$, $\beta_{\text{ent}} = 0.01$

Output: Updated Actor and Critic parameters

```

1: for  $i = 1, \dots, N_{\text{opt}}$  do
2:    $(S_t, \mathbf{a}, r_t, S_{t+1}) \leftarrow \mathcal{B}.\text{sample}(64)$ 
3:    $\text{target} \leftarrow r_t + \gamma \cdot \text{CRITIC}(S_{t+1})$  ▷ TD bootstrap: critic estimates own target
4:    $\mathcal{L}_C \leftarrow \text{MSE}(\text{CRITIC}(S_t), \text{target});$  update Critic
5:    $\ell_g \leftarrow \text{ACTOR}(S_t);$   $\log \pi \leftarrow a_g \log \sigma(\ell) + (1-a_g) \log \sigma(-\ell)$ 
6:    $A_t \leftarrow \text{target} - \text{CRITIC}(S_t)$ 
7:    $\mathcal{L}_\pi \leftarrow -(\log \pi \cdot A_t).\text{mean} - \beta_{\text{ent}} \cdot H$  ▷ Per-gene Bernoulli PG
8:   Update Actor; clip gradients at 1.0
9: end for

```

Algorithm 2b: Explore Best — Logit-space CEM with rank-based elite weights

Input: Population rewards & panels from explore(), logits ℓ , adaptive α , ρ

Output: Updated logits ℓ and probabilities \mathbf{p}

```

1: Sort population by reward;  $E \leftarrow \lceil \rho \cdot M \rceil$  elites ( $\rho$  adaptive by spread)
2:  $w_i \leftarrow \exp(-\text{rank}_i / \tau_r)$ ; normalize ▷ Rank-based weighting
3:  $\text{freq}_g \leftarrow \text{np.bincount}(\text{flat\_idx}, \text{weights} = w_{\text{rep}})$ ; clip; normalize
4:  $\hat{\ell}_g \leftarrow \log(\text{freq}_g)$ 
5: for  $i = 1, \dots, N_{\text{opt}}$  do
6:    $\ell_g \leftarrow (1 - \alpha) \ell_g + \alpha \hat{\ell}_g$  ▷ Logit-space EMA
7: end for
8:  $p_g \leftarrow \text{softmax}(\ell)$ ; clip  $\geq p_{\min}$ ; renormalize
9:  $\ell_g \leftarrow \log(p_g)$  ▷ Re-sync logits

```

Algorithm 2c: Optimize Best — KL-regularized CEM with trust-region backtracking

Input: Buffer \mathcal{B} , panel logits ℓ , elite frac ρ , step size η , KL target δ

Output: Updated logits ℓ

```

1: for  $i = 1, \dots, N_{\text{opt}}$  do
2:    $(S_t, \mathbf{a}, r_t, \dots) \leftarrow \mathcal{B}.\text{sample}(64)$ 
3:    $r \leftarrow (r - \bar{r}) / (\sigma_r + \varepsilon)$  ▷ Standardize
4:    $E \leftarrow \lceil \rho \cdot |\text{batch}| \rceil$  elites by reward
5:    $w_i \leftarrow \text{softmax}(r_i / 0.75)$  ▷ Magnitude-based weights
6:    $m_g \leftarrow \sum_i w_i \cdot a_{ig}$  ▷ Weighted marginals
7:    $m \leftarrow (1 - \lambda) m + \lambda p_0$  ▷ Anti-collapse mixing;  $p_0 = K/G$ 
8:    $p_{\text{old}} \leftarrow \sigma(\ell);$   $\eta_0 \leftarrow \eta$ 
9:   for  $\text{bt} = 1, \dots, 20$  do ▷ Trust-region backtracking
10:     $p_{\text{new}} \leftarrow (1 - \eta) p_{\text{old}} + \eta m$ 
11:     $\text{KL} \leftarrow \text{mean}(p_{\text{old}} \log \frac{p_{\text{old}}}{p_{\text{new}}} + (1 - p_{\text{old}}) \log \frac{1 - p_{\text{old}}}{1 - p_{\text{new}}})$ 
12:    if  $\text{KL} \leq \delta$  then break
13:    else  $\eta \leftarrow \eta \times 0.5$ 
14:    end if
15:  end for
16:   $\ell \leftarrow \text{logit}(\text{clamp}(p_{\text{new}}));$   $\tau_{\text{sample}} \leftarrow \max(\tau_{\text{end}}, \tau \times 0.98)$ 
17: end for

```

3. Summary of Key Algorithmic Differences

Component	Original	Explore Best	Optimize Best
Paradigm	RL (actor-critic)	CEM + logit distribution	CEM + logit distribution
Sampling	ε -greedy + \mathcal{N} noise	Gumbel-TopK + mutation + prior-preserve	Gumbel-TopK (single strategy)
Screening	None (direct eval)	Linear surrogate UCB ($O(K)/\text{cand}$)	Neural surrogate MC-Dropout UCB
Novelty	None	Jaccard filter (adaptive θ)	None
Temp. control	$\varepsilon \times 0.97$ (monotonic)	Adaptive kick: $\times 1.5/\times 0.9$ by spread	Monotonic anneal $\times 0.98$
Update	Policy gradient + TD(0) critic	Rank-weighted logit-space EMA	KL-regularized trust-region
Weights	Advantage A_t	$e^{-\text{rank}/\tau}$ (scale-free)	$\text{softmax}(r/0.75)$ (magnitude)
Reward	$\alpha \cdot \text{ARI} + (1-\alpha) \cdot s$	$(0.5A + 0.3N + 0.2S)^{1.5} - \text{pen}$	$0.45f(A) + 0.45f(N) + 0.1f(S) - \lambda s_p$
Memo.	None	Sorted-idx FIFO cache	Sorted-index FIFO cache
Robustness	Undefined logger	Gene universe sanitization	No sanitization

4. Performance Comparison

Metric	Original	Explore Best	Optimize Best
Evolution score	0.500	1.000	0.958
Best ARI	—	0.513	0.646
Best NMI	—	0.621	0.672
Training speed (best)	—	0.825	0.640
Panel size	500	500–548	500
Iterations to best	—	27	10
Post-best stagnation	—	22	39
Success rate	—	74%	96%
Improvement rate	—	26%	20%
LLM cost	—	\$5.41	\$6.29

5. Convergent Evolution: Why Both Abandoned RL for CEM

Shared insight: RL → CEM paradigm shift (independently discovered by both runs)

- **Objective–architecture mismatch:** The original RL treats panel selection as a *sequential decision* (per-gene Bernoulli actions), but reward depends only on the *final set*. TD bootstrapping estimates intermediate state values that have no meaningful interpretation, introducing harmful bias. Both runs eliminated the critic and TD targets entirely.
- **Combinatorial vs. sequential:** Selecting 500 from 2000 genes is a combinatorial optimization problem ($\binom{2000}{500}$ candidates). CEM samples and evaluates *complete panels*, directly optimizing over the combinatorial search space rather than decomposing it into G independent binary decisions.
- **Multi-metric reward:** Both runs independently evolved from single-metric (ARI only) to multi-metric (ARI + NMI + SI) rewards with nonlinear shaping, providing a richer optimization signal.
- **Evaluation memoization:** Both discovered sorted-index-tuple caching to avoid redundant expensive clustering evaluations — a natural consequence of population-based search encountering repeated candidates.

6. Why Each Variant Improved

Explore Best: Superior search escape → higher final score (1.0)

- **Adaptive temperature kick** (spread-based): When the population's reward spread drops below 0.05 (stagnation), T is multiplied by 1.5, forcing broader sampling. This broke through the 0.956 plateau at iteration 27 to reach 1.0. The Optimize variant's monotonic τ decay ($\times 0.98/\text{epoch}$) cannot recover once converged.
- **Jaccard novelty filter**: Explicitly rejects candidates too similar to best/elites (adaptive threshold 0.90–0.98). This prevents the search from wasting evaluation budget on near-duplicate panels and forces genuine exploration of new gene combinations.
- **Three-strategy sampling**: Global Gumbel-TopK, local swap mutation (probability-biased removal + Gumbel-top1 addition), and prior-preserving sampling cover different scales of variation. The Optimize variant uses only global sampling.
- **Lightweight surrogate**: Linear ridge with diagonal posterior ($O(K)$ per candidate) enables screening $6\times$ more candidates than the neural surrogate, maximizing search breadth within the evaluation budget.
- **Gene universe sanitization**: Deduplication and intersection with `adata.var_names` prevents silent failures from duplicate or missing genes, improving robustness across diverse evaluation conditions.

Optimize Best: Superior update stability → faster early convergence (0.958 by iter 10)

- **KL trust-region** guarantees bounded updates: Bernoulli $\text{KL}(p_{\text{old}} \| p_{\text{new}}) \leq \delta$ with backtracking prevents overshooting. This yields 96% success rate (vs. Explore's 74%) and reaches 0.958 by iteration 10 (vs. Explore's iteration 27 for comparable scores).
- **Anti-collapse prior mixing**: Blending elite marginals with uniform $p_0 = K/G$ (decay rate tied to ε) prevents the distribution from collapsing to a point mass too early, maintaining diversity in early epochs when information is scarce.
- **Neural surrogate with MC-Dropout UCB**: The embedding-based surrogate captures nonlinear gene interactions that the linear model cannot. MC-Dropout uncertainty provides calibrated exploration bonuses, directing evaluation to high-potential but uncertain candidates.
- **Elitism (always evaluate incumbent)**: Guarantees the best-known panel is always scored, preventing regression due to stochastic evaluation noise and ensuring monotonic best-score tracking.
- **Limitation — no escape mechanism**: The monotonic temperature decay ($\times 0.98/\text{epoch}$) and lack of novelty filtering caused 39 iterations of stagnation after reaching 0.958. The distribution converged too early, and the trust-region update (designed for stability) prevented the large jumps needed to escape the local optimum.