

Algorithmic Flow: Soft Cluster Assignment (update_R)

Three evolved strategies for batch-balanced cell-cluster responsibility assignment

Original Harmony Block-wise heuristic

1. Soft k-means base:
 $S_{ki} = \exp(-D_{ki}/\sigma_k)$; $S \leftarrow S / \sum_k S$

2. Random shuffle cells:
 $\pi \leftarrow \text{RANDPERM}(N)$

3. Split into ~ 20 blocks
(block_size = 5% of N)

4. **For each block \mathcal{B} :**
1. Remove block from O, E
2. ratio = $(E/(O+E))^\theta$
Heuristic penalty
3. $R_{\mathcal{B}} = S_{\mathcal{B}} \odot (\text{ratio} \cdot \Phi_{\mathcal{B}})$
4. Normalize per cell
5. Add block back to O, E

5. Restore order via π^{-1}

Sequential block updates;
order-dependent results

harmony_272 (IPF) Iterative Proportional Fitting

1. Stable softmax base:
 $R = \text{SOFTMAX}(-D/\sigma, \text{dim}=0)$
No block shuffle; global op

2. Anneal mixing ρ :
 $\rho = (1-t)\rho_{\text{start}} + t\rho_{\text{end}}$
High early \rightarrow low later

3. Adaptive IPF iterations:
 $\Delta R > 0.15 \Rightarrow$ fewer iters
 $\Delta R < 0.05 \Rightarrow$ more iters

4. **For $j = 1 \dots n_{\text{ipf}}$:**
1. $O = R\Phi^\top$; $E = \text{outer}(\sum R, \text{Pr}_b)$
2. $T = (1-\rho)O + \rho E$
Soft batch-balance target
3. $S = T/O$; $G = S\Phi$ (scales)
4. $R \leftarrow R \odot G$; renormalize
(in-place, pre-allocated buf)

5. Update O, E from final R

harmony_best_mixing (OT) Entropic Optimal Transport

1. Anneal regularization:
 $\varepsilon = \varepsilon_{\text{max}} + (\varepsilon_{\text{min}} - \varepsilon_{\text{max}})t$
Smooth early \rightarrow sharp later

2. Diversity penalty:
 $\theta_{\log} = \theta \cdot \log(1 + O/E)$

3. **For each batch b :**
Cost $C = D_{:, \mathcal{I}_b} / \sigma$
Supply $a = E_{:, b}$ (cluster mass)
Demand $u = \mathbf{1}$ (uniform)

4. Sinkhorn (warm-start):
1. $\log K = -C/\varepsilon$
2. **For $\ell = 1 \dots L_{\text{max}}$:**
 $\log \mathbf{u} \leftarrow \log a - \text{LSE}(\log K + \log \mathbf{v}^\top)$
 $\log \mathbf{v} \leftarrow \log u - \text{LSE}(\log K^\top + \log \mathbf{u})$
Hard OT constraint
3. If converged: **break**

5. $P = \exp(\log \mathbf{u} + \log K + \log \mathbf{v}^\top)$
 $R_{:, \mathcal{I}_b} \leftarrow P$; update O, E

Algorithmic Flow: Batch Correction (moe_correct_ridge)

Three evolved strategies for ridge-regression-based batch effect removal

Original Harmony

Per-cluster matrix inverse

1. Clone original data:
 $Z_{\text{corr}} \leftarrow Z_{\text{orig}}.\text{clone}()$

2. For $k = 1 \dots K$:

- $\Phi_{Rk} = \Phi_{\text{moe}} \odot R[k, :]$
- $\text{Cov} = \Phi_{Rk} \Phi_{\text{moe}}^\top + \text{diag}(\lambda)$
- $\text{Cov}^{-1} = \text{INV}(\text{Cov})$
Full inverse $\times K$ clusters
- $W = \sum_b \text{Cov}_{:,b}^{-1} (\sum_{i \in b} R_{k,i} Z_i)^\top$
- $W[0, :] = 0$ (keep intercept)
- $Z_{\text{corr}} \text{ -= } W^\top \Phi_{Rk}$

3. Cosine embedding:
 $Z_{\text{cos}} = Z_{\text{corr}} / \|Z_{\text{corr}}\|_2$

No correction control —
full correction applied directly.
No regularization beyond λ .

harmony_272 (IPF)

Cholesky + trust-region

1. Sufficient statistics $S_{\Phi Z}$
via contiguous batch GEMMs
(exploit one-hot Φ , pre-sorted)

2. Assemble $A \in \mathbb{R}^{K \times (B+1) \times (B+1)}$
 $L = \text{BATCHCHOLESKY}(A)$
 $W = \text{CHOLSOLVE}(S_{\Phi Z}, L)$
Single batched solve

3. Tie-to-mean regularization:
 $W \leftarrow (W + \gamma \bar{W}) / (1 + \gamma)$
 γ adaptive per cluster; $W[:, 0, :] = 0$

4. Correction via per-batch GEMMs:
 $\text{corr}_{:, \mathcal{I}_b} = W_{:, b+1}^\top R_{:, \mathcal{I}_b}$

5. Trust-region backtracking:

- $\eta \leftarrow \eta_{\text{init}}$
- $Z_{\text{cand}} = Z_{\text{orig}} - \eta \cdot \text{corr}$
- Eval: $\text{obj}_{\text{new}}, R_{\text{cand}}, \Delta R$
- Accept if $\text{obj}_{\text{new}} \leq \text{obj}_{\text{old}}$ and $\|\Delta R\| \leq \delta$
Prevents overcorrection
- Else $\eta \leftarrow \eta/2$; up to $4\times$

6. Commit or fallback to η_{min}

harmony_best_mixing (OT)

Schur complement + trust region

1. Compute $S_{\Phi Z}$ via
pre-permuted batch-slice GEMMs

2. Schur complement solve
(no inverse/Cholesky needed):
 $D_{\text{inv}} = 1 / (O + \lambda_{\text{batch}})$ (diag)
 $s_{\text{schur}} = s_k - \sum_b o_{kb}^2 D_{\text{inv}}$
 $w_0 = \text{rhs}_0 / s_{\text{schur}}$
 $w_b = D_{\text{inv}} (b_b - o_b w_0)$
Closed-form; $O(KB)$ vs $O(KB^3)$

3. Cluster-adaptive shrinkage:
 $\gamma_k = \gamma_0 \bar{s} / s_k$
 $W \leftarrow (W + \gamma_k \bar{W}) / (1 + \gamma_k)$
Stronger for smaller clusters

4. Correction via per-batch-slice
GEMMs (contiguous memory)

5. Per-cell trust region:
 $\rho = \rho_{\text{min}} + (\rho_{\text{max}} - \rho_{\text{min}}) t$
If $\|\text{corr}_i\| > \rho \|Z_i\|$: rescale to boundary
Per-cell overcorrection guard;
 ρ anneals: conservative \rightarrow permissive

6. $Z_{\text{corr}} = Z_{\text{orig}} - \text{corr}$
 $Z_{\text{cos}} = Z_{\text{corr}} / \|Z_{\text{corr}}\|_2$

Detailed Pseudocode Comparison

Algorithmic differences in soft assignment (UPDATE_R) and batch correction (MOE_CORRECT_RIDGE)

1. Soft Cluster Assignment (update_R)

Algorithm 1a: Original Harmony — Block-wise Heuristic Update

Input: Distance matrix $D \in \mathbb{R}^{K \times N}$, bandwidth $\sigma \in \mathbb{R}^K$, batch indicator $\Phi \in \{0, 1\}^{B \times N}$

Output: Updated responsibility matrix $R \in \mathbb{R}^{K \times N}$

- 1: $S_{ki} \leftarrow \exp(-D_{ki}/\sigma_k)$; $S \leftarrow S / \sum_k S$ ▷ Soft k-means base
- 2: $\pi \leftarrow \text{RANDPERM}(N)$ ▷ Random shuffle
- 3: Split $\{1, \dots, N\}$ into blocks of size $\lfloor N \cdot \text{block_size} \rfloor$
- 4: **for** each block \mathcal{B} **do**
- 5: $O \leftarrow O - R_{:, \mathcal{B}} \Phi_{:, \mathcal{B}}^\top$; $E \leftarrow E - \text{outer}(\sum_{\mathcal{B}} R, \text{Pr}_b)$ ▷ Remove block
- 6: $\text{ratio} \leftarrow \left(\frac{E}{O + E} \right)^\theta$ ▷ Heuristic diversity penalty
- 7: $R_{:, \mathcal{B}} \leftarrow S_{:, \mathcal{B}} \odot (\text{ratio} \cdot \Phi_{:, \mathcal{B}})$; normalize per cell
- 8: $O \leftarrow O + R_{:, \mathcal{B}} \Phi_{:, \mathcal{B}}^\top$; $E \leftarrow E + \text{outer}(\sum_{\mathcal{B}} R, \text{Pr}_b)$ ▷ Add back
- 9: **end for**
- 10: Restore original cell order via π^{-1}

Algorithm 1b: harmony_272 — Iterative Proportional Fitting (IPF)

Input: Distance matrix D , bandwidth σ , batch indicator Φ , Harmony iteration i/T

Output: Updated R

- 1: $R \leftarrow \text{SOFTMAX}(-D/\sigma, \text{dim} = 0)$ ▷ Stable softmax (no block shuffle)
- 2: $\rho \leftarrow (1 - t) \cdot \rho_{\text{start}} + t \cdot \rho_{\text{end}}$ where $t = (i - 1)/(T - 1)$ ▷ Annealing schedule
- 3: Adapt IPF iterations: $n_{\text{ipf}} \leftarrow f(\|\Delta R\|)$ ▷ Fewer if unstable, more if stable
- 4: **for** $j = 1, \dots, n_{\text{ipf}}$ **do**
- 5: $O \leftarrow R \Phi^\top$; $E \leftarrow \text{outer}(\sum_i R, \text{Pr}_b)$
- 6: $T_{\text{target}} \leftarrow (1 - \rho) \cdot O + \rho \cdot E$ ▷ Soft batch-balance target
- 7: $S \leftarrow T_{\text{target}} / O$; $G \leftarrow S \Phi$ ▷ Per-cell scaling factors
- 8: $R \leftarrow R \odot G$; $R \leftarrow R / \sum_k R$ ▷ Multiplicative reweighting
- 9: **end for**
- 10: Update O, E from final R

Algorithm 1c: harmony_best_mixing — Per-batch Entropic Optimal Transport (Sinkhorn)

Input: Distance matrix D , bandwidth σ , batch indicator Φ , Harmony iteration i/T

Output: Updated R

- 1: $\varepsilon \leftarrow \varepsilon_{\text{max}} + (\varepsilon_{\text{min}} - \varepsilon_{\text{max}}) \cdot t$ ▷ Entropic regularization annealing
- 2: **for** each batch $b = 1, \dots, B$ **do**
- 3: $C \leftarrow D_{:, \mathcal{I}_b} / \sigma$ ▷ Cost matrix for batch b cells
- 4: $a \leftarrow E_{:, b} \cdot (n_b / \sum E_{:, b})$ ▷ Cluster supply (desired mass)
- 5: $u \leftarrow \mathbf{1}_{n_b}$ ▷ Cell demand (uniform)
- 6: Initialize $\log \mathbf{u}, \log \mathbf{v}$ from warm-start duals
- 7: $\log K \leftarrow -C/\varepsilon$
- 8: **for** $\ell = 1, \dots, L_{\text{max}}$ **do** ▷ Log-domain Sinkhorn iterations
- 9: $\log \mathbf{u} \leftarrow \log a - \text{LSE}_{\text{col}}(\log K + \log \mathbf{v}^\top)$
- 10: $\log \mathbf{v} \leftarrow \log u - \text{LSE}_{\text{row}}(\log K + \log \mathbf{u})$
- 11: **if** $\max(|\Delta \log \mathbf{u}|, |\Delta \log \mathbf{v}|) < \tau$ **then break**
- 12: **end if**
- 13: **end for**
- 14: $P \leftarrow \exp(\log \mathbf{u} + \log K + \log \mathbf{v}^\top)$; normalize columns
- 15: $R_{:, \mathcal{I}_b} \leftarrow P$ ▷ Assign transport plan to R
- 16: **end for**
- 17: $O \leftarrow \text{SCATTERADD}(R, \text{batch_id})$; $E \leftarrow \text{outer}(\sum R, \text{Pr}_b)$

2. Batch Effect Correction (moe_correct_ridge)

Algorithm 2a: Original Harmony — Per-cluster Matrix Inverse

Input: $Z_{\text{orig}} \in \mathbb{R}^{d \times N}$, $R \in \mathbb{R}^{K \times N}$, $\Phi_{\text{moe}} \in \mathbb{R}^{(B+1) \times N}$, $\lambda \in \mathbb{R}^{B+1}$

Output: Z_{corr} , updated Z_{cos}

```

1:  $Z_{\text{corr}} \leftarrow Z_{\text{orig}}.\text{CLONE}()$ 
2: for  $k = 1, \dots, K$  do
3:    $\Phi_{Rk} \leftarrow \Phi_{\text{moe}} \odot R[k, :]$ 
4:    $\text{Cov} \leftarrow \Phi_{Rk} \Phi_{\text{moe}}^\top + \text{diag}(\lambda)$ 
5:    $\text{Cov}^{-1} \leftarrow \text{INV}(\text{Cov})$  ▷ Full  $(B+1) \times (B+1)$  inverse per cluster
6:    $W \leftarrow \sum_{b=0}^B \text{Cov}_{:,b}^{-1} \cdot \left( \sum_{i \in \text{batch}_b} R_{k,i} Z_{:,i} \right)^\top$ 
7:    $W[0, :] \leftarrow 0$  ▷ Keep intercept
8:    $Z_{\text{corr}} \leftarrow Z_{\text{corr}} - W^\top \Phi_{Rk}$ 
9: end for
10:  $Z_{\text{cos}} \leftarrow Z_{\text{corr}} / \|Z_{\text{corr}}\|_2$ 

```

Algorithm 2b: harmony_272 — Batched Cholesky + Trust-region Backtracking

Input: Z_{orig} , R , Φ , batch pointers, Harmony iteration i/T

Output: Z_{corr} , updated Z_{cos} , R , O , E

```

1: Compute  $S_{\Phi Z}^{(k,b)} \leftarrow \sum_{i \in \text{batch}_b} R_{k,i} Z_{:,i}$  via contiguous batch GEMMs
2: Assemble  $A \in \mathbb{R}^{K \times (B+1) \times (B+1)}$  exploiting one-hot  $\Phi$  structure
3:  $L \leftarrow \text{BATCHEDCHOLESKY}(A)$ ;  $W \leftarrow \text{CHOLESKYSOLVE}(S_{\Phi Z}, L)$  ▷ Replaces  $K$  separate inverses
4:  $W[:, 0, :] \leftarrow 0$ 
5:  $W \leftarrow (W + \gamma \bar{W}) / (1 + \gamma)$  ▷ Tie-to-mean regularization
6: Compute correction via per-batch GEMMs:  $\text{corr}_{:, \mathcal{I}_b} \leftarrow W_{:, b+1}^\top R_{:, \mathcal{I}_b}$ 
7:  $\eta \leftarrow \eta_{\text{init}}$  ▷ Step size for line search
8: for  $j = 1, \dots, n_{\text{backtrack}}$  do ▷ Trust-region backtracking
9:    $Z_{\text{cand}} \leftarrow Z_{\text{orig}} - \eta \cdot \text{corr}$ 
10:  Evaluate candidate:  $\text{obj}_{\text{new}}, R_{\text{cand}} \leftarrow \text{CANDIDATEEVAL}(Z_{\text{cand}})$ 
11:  if  $\text{obj}_{\text{new}} \leq \text{obj}_{\text{old}}$  and  $\|\Delta R\|_1 / N \leq \delta_{\text{thresh}}$  then
12:    Accept: commit  $Z_{\text{corr}}, Z_{\text{cos}}, R, O, E$  ▷ Objective must improve
13:    break
14:  end if
15:   $\eta \leftarrow \eta / 2$ 
16: end for
17: If not accepted: apply conservative step with  $\eta_{\text{min}}$ 

```

Algorithm 2c: harmony_best_mixing — Schur Complement Solve + Per-cell Trust Region

Input: Z_{orig} , R , batch slices, Harmony iteration i/T

Output: Z_{corr} , updated Z_{cos}

```

1: Compute  $S_{\Phi Z}$  via contiguous batch-slice GEMMs (pre-permuted data)
2:  $D_{\text{inv}} \leftarrow 1 / (O + \lambda_{\text{batch}})$  ▷ Diagonal inverse of batch block
3:  $s_{\text{schur}} \leftarrow s_k - \sum_b o_b^2 \cdot D_{\text{inv}, kb}$  ▷ Schur complement (no inverse/Cholesky)
4:  $w_0 \leftarrow (b_0 - \sum_b o_b \cdot D_{\text{inv}, b} \cdot b_b) / s_{\text{schur}}$ 
5:  $w_b \leftarrow D_{\text{inv}, b} \cdot (b_b - o_b \cdot w_0)$ 
6: Pack  $W = [w_0; w_1; \dots; w_B]$ ;  $W[:, 0, :] \leftarrow 0$ 
7:  $\gamma_k \leftarrow \gamma_0 \cdot \bar{s} / s_k$ ;  $W \leftarrow (W + \gamma_k \bar{W}) / (1 + \gamma_k)$  ▷ Cluster-size-adaptive shrinkage
8: Compute correction via per-batch GEMMs
9:  $\rho \leftarrow \rho_{\text{min}} + (\rho_{\text{max}} - \rho_{\text{min}}) \cdot t$  ▷ Annealed trust radius
10: for each cell  $i$  do
11:   if  $\|\text{corr}_i\|_2 > \rho \cdot \|Z_i\|_2$  then ▷ Per-cell magnitude constraint
12:      $\text{corr}_i \leftarrow \text{corr}_i \cdot \rho \|Z_i\| / \|\text{corr}_i\|$  ▷ Rescale to trust boundary
13:   end if
14: end for
15:  $Z_{\text{corr}} \leftarrow Z_{\text{orig}} - \text{corr}$ ;  $Z_{\text{cos}} \leftarrow Z_{\text{corr}} / \|Z_{\text{corr}}\|_2$ 

```

3. Summary of Key Algorithmic Differences

Component	Original		harmony_272 (IPF)	harmony_best_mixing (OT)
Assignment strategy	Block-wise $(E/(O+E))^\theta$ penalty	heuristic: ratio	Global softmax + IPF multiplicative reweighting toward target T	Per-batch entropic OT (Sinkhorn) with warm-started duals
Batch balance	Soft penalty via θ -powered ratio		Interpolation target $T = (1-\rho)O + \rho E$ with annealed ρ	Hard marginal constraint via OT (supply = $E_{:,b}$)
Ridge solve	K separate matrix inverses		Single batched Cholesky solve	Closed-form Schur complement (no factorization)
Correction control	None (full correction applied directly)		Backtracking line search: requires obj \downarrow and $\ \Delta R\ \leq \delta$	Per-cell trust region: $\ \text{corr}_i\ \leq \rho \ Z_i\ $
Regularization	Diagonal ridge λ only		Ridge + tie-to-mean (γ adaptive per cluster size)	Ridge + tie-to-mean ($\gamma_k \propto \bar{s}/s_k$)
Annealing	None		ρ : mixing strength decreases over iterations	ε : OT sharpness increases; ρ : trust region grows
Data layout	Random permutation per call		Pre-sorted by batch; contiguous block pointers	Pre-permuted tensors; batch slices (s_b, e_b)

4. Performance Comparison

Metric	Original	harmony_272	harmony_best_mixing	Best
Batch mixing score	0.7094	0.7118 (+0.3%)	0.7842 (+10.5%)	OT
Bio conservation score	0.6777	0.7063 (+4.2%)	0.6464 (-4.6%)	IPF
Speed score	0.1256	0.8598 ($\times 6.8$)	0.0541 (-56.9%)	IPF
Convergence score	0.8312	0.5527 (-33.5%)	0.7572 (-8.9%)	Orig
Evolution generation	0	6	8	—
Evolution order	0	272	277	—
Mutation category	—	implementation	optimization_method	—

5. Why Each Variant Improved

harmony_272: Balanced improvement via IPF + trust-region

- **Speed ($\times 6.8$)**: Replaced $O(N)$ block-wise random shuffle + sequential block updates with a single global softmax followed by ~ 5 in-place IPF iterations using pre-allocated buffers. Eliminated per-call random permutations and reduced memory allocation overhead.
- **Bio conservation (+4.2%)**: Trust-region backtracking prevents overcorrection by requiring both (i) objective decrease and (ii) stable responsibilities ($\|\Delta R\| \leq \delta$) before accepting a correction step. Falls back to conservative step size when criteria are not met.
- **Mixing (+0.3%)**: Modest gain from annealed ρ schedule that applies stronger batch-balance pressure in early iterations when batch effects dominate, and relaxes later to preserve biological structure.

harmony_best_mixing: Maximum mixing via entropic optimal transport

- **Mixing (+10.5%)**: Replaced heuristic ratio penalty with per-batch entropic OT that enforces batch-balanced cluster composition as a hard marginal constraint (supply = $E_{:,b}$). The Sinkhorn algorithm guarantees feasibility of the transport plan, ensuring every cluster receives its expected share of cells from each batch.
- **Bio conservation (-4.6%)**: Trade-off from stronger batch mixing enforcement. Partially mitigated by per-cell trust region ($\|\text{corr}_i\| \leq \rho \|Z_i\|$) and cluster-size-adaptive regularization ($\gamma_k \propto 1/s_k$), which prevents small clusters from being over-corrected.
- **Speed (-56.9%)**: Log-domain Sinkhorn with up to 25 iterations per batch is computationally expensive. Warm-starting duals reduces average iterations in practice, but the per-batch loop remains the bottleneck.