TESTS REVEAL AN UNSTATED CODE: $(N{+}1)$ PROPAGATION CONSISTENCY FOR LLM CODE GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Selecting LLM-generated code without ground-truth tests relies on LLM-generated test inputs and expected outputs. Yet whether the LLM-generated tests' expected outputs or the candidate codes are the more reliable signal varies sharply across tasks. Existing methods either trust the LLM-generated tests' expected outputs as ground truth or discard them in favor of cross-code consensus. To our knowledge, no prior framework unifies these two stances as endpoints of a principled, task-adaptive continuum; existing combinations remain heuristic. Our unifying insight: *the LLM-generated tests' expected outputs are the input-output behavior of an unstated $(n{+}1)$-th candidate code implicit in the LLM's solution distribution, on equal footing with the n explicit ones.* Each of the $n{+}1$ codes defines an agreement view, and we identify codes that score consistently across all $n{+}1$ views. We propose (**N**+1)-**P**ropagation **C**onsistency (**N$^+$PC**), which couples $n{+}1$ Laplacian propagations via a task-adaptive scalar κ weighting the unstated-code view against the rest. Its boundary cases recover a reference-trusting scorer and reference-free graph aggregation, connecting both information regimes within one objective. As a plug-and-play framework, N$^+$PC consistently improves all inner baselines and advances state-of-the-art Pass@k among execution-only rerankers.

1 INTRODUCTION

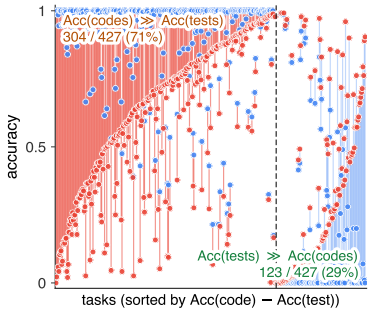
Code generation with large language models (LLMs) is now widely deployed (Chen et al., 2021; Li et al., 2022). Because any individual sample is frequently incorrect, a common test-time scaling strategy samples many candidates and ranks them to surface those most likely correct (Chen et al., 2023; Wu et al., 2025; Snell et al., 2025). Selection requires tests; without ground-truth tests, the LLM also generates the test inputs together with their expected outputs. The selector then judges each candidate by how its outputs align with these expected outputs.

Existing methods split into two camps that take opposite stances on the LLM-generated expected outputs. One trusts them as ground truth and ranks candidates by per-test agreement, including CodeT (Chen et al., 2023), ACES (Sun et al., 2026), and B^4 (Chen et al., 2024a). The other discards them entirely and relies on cross-code execution agreement, including MBR-exec (Shi et al., 2022) and SRank (To et al., 2024). Both camps therefore rest on a strong, opposing implicit assumption about the same object: the reliability of the LLM-generated expected outputs.

Yet the relative reliability of candidate codes and LLM-generated tests varies sharply across tasks. On MBPP (Austin et al., 2021), we measure per task the correctness rates of candidate codes and LLM-generated tests' expected outputs (Fig. 1, left): codes prevail on a substantial fraction of tasks (red segments), tests on the remainder (blue). Where the codes are the more reliable signal, methods trusting the expected outputs inherit that unreliability; Fig. 1 (right) shows extreme cases where every LLM-generated test has a wrong expected output despite most candidates being correct. Where the LLM-generated tests are the more reliable signal, methods relying on cross-code consensus alone forgo the very evidence that would inform the ranking. To our knowledge, no prior framework unifies the two stances as endpoints of a single principled, task-adaptive continuum; existing combinations remain heuristic.

This unification hinges on an observation that prior framings overlook. Each LLM-generated expected output is the LLM's best guess at what a correct code would produce on the corresponding

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.



MBPP/422: Write a python function to find the average of cubes of first n natural numbers.

```
def find_Average_Of_Cube(n):
  s = sum(i**3 for i in range(n+1))
  return s/n
```

input	LLM expected	true output
n=3	9.0 ✗	12.0 ✓
n=4	20.0 ✗	25.0 ✓
n=6	91.0 ✗	73.5 ✓

MBPP/770: Write a python function to find the sum of fourth power of first n odd natural numbers.

```
def odd_Num_Sum(n):
  s = sum(k**4 for k in range(1,2*n,2))
  return s
```

input	LLM expected	true output
n=2	17 ✗	82 ✓
n=3	98 ✗	707 ✓
n=4	354 ✗	3108 ✓

Figure 1: **Candidate codes and LLM-generated tests trade reliability across tasks: neither side is universally the more reliable signal.** *Left:* per-task code accuracy (•, $n{\approx}200$ candidates) versus LLM-generated test accuracy (•, $m{\approx}500$ tests), sorted by their gap; red segments mark code>test, blue the inverse. *Right:* two extreme tasks where most candidates are correct yet every LLM-generated test has a wrong expected output. Generator: GPT-3.5-Turbo on MBPP throughout.

test input, so the m tests' expected outputs jointly trace out the input-output behavior of an additional code implicit in the LLM's solution distribution. *This unstated code is the $(n{+}1)$-th candidate code of the problem, on equal footing with the n explicit ones.* Once this unstated code joins its explicit peers, the binary contrast dissolves into a single question: how should these $n{+}1$ same-kind codes be combined to rank the explicit candidates?

This reframe immediately yields a mechanism. Each of the $n{+}1$ codes defines an agreement view by comparing its outputs with those of the n explicit candidates on each test. Each view is an $n \times m$ matrix, with the reference code's own row zeroed when it is an explicit candidate. A pass-matrix reranker then gives each view a score vector; we seek a ranking consistent across these views. We propose (**N**+1)-**P**ropagation **C**onsistency (**N$^+$PC**), which couples $n{+}1$ Laplacian propagations (one per view) via a task-adaptive scalar κ calibrating the unstated-code view's weight against the rest.

N$^+$PC connects the two stances through distinct boundary configurations. At $\kappa \to \infty$ with $\lambda = 0$, it recovers the inner scorer on the LLM-generated expected outputs alone, including ACES, CodeT, and MV. At $\kappa = 0$, it performs reference-free graph aggregation using only cross-code agreement. Between these regimes, the task-adaptive κ fuses both signals. N$^+$PC is therefore a principled unification rather than a heuristic mixture, evaluated on HumanEval (Chen et al., 2021), HumanEval$^+$ (Liu et al., 2023), and MBPP.

Our main contributions are as follows.

- **A unifying continuum view of two opposing stances on LLM-generated tests.** We connect both information regimes as endpoints of a principled, task-adaptive continuum.
- **The unstated-code reframe and the N$^+$PC framework.** We recast the LLM expected outputs as an unstated $(n{+}1)$-th code, then propose N$^+$PC, which couples $n{+}1$ Laplacian propagations via task-adaptive κ to combine reference-based and cross-code evidence.
- **Strong, plug-and-play empirical gains.** N$^+$PC is inner-agnostic: instantiated with MV, ACES-C, and ACES-O on HumanEval, HumanEval$^+$, and MBPP, it consistently improves every inner baseline and advances state-of-the-art Pass@k among execution-only rerankers.

2 Preliminaries and Motivation

Treating the unstated code as the $(n{+}1)$-th candidate (§1) lifts the classical pass matrix to a *generalized pass matrix* $\mathcal{B}$ that records all pairwise per-test agreements among the $n{+}1$ codes. We show that each prior family reads only a *strict subset* of $\mathcal{B}$, motivating a framework that reads it in full.

2.1 SETUP: FROM PASS MATRIX TO $\mathcal{B}$ VIA THE UNSTATED CODE

Pass matrix. We adopt the standard post-hoc reranking setup with LLM-generated tests (Sun et al., 2026; Chen et al., 2023). For a single problem, an LLM produces n candidate programs and m tests:

$$\{c_k\}_{k=1}^{n}, \qquad \{t_j\}_{j=1}^{m} \text{ with } t_j = (x_j, \hat{y}_j), \tag{1}$$

where x_j is the input and $\hat{y}_j$ is the LLM-generated expected output. Executing each candidate on each input and comparing the result to $\hat{y}_j$ yields the familiar *pass matrix*

$$B \in \{0,1\}^{n\times m}, \qquad B_{k,j} \;=\; \mathbf{1}[\mathrm{exec}(c_k, x_j) = \hat{y}_j], \tag{2}$$

with a unique sentinel for compile errors, runtime exceptions, timeouts, and unparseable outputs.

Unstated code and generalized pass matrix. The m LLM-generated expected outputs $\{\hat{y}_j\}_{j=1}^{m}$ trace the input-output behavior of an implicit function: the *unstated code* of the LLM's solution distribution. We denote this virtual $(n{+}1)$-th candidate by c_{n+1} and gather all $n{+}1$ candidates' outputs in a single $(n{+}1) \times m$ *output matrix*

$$Y_{k,j} \;=\; \mathrm{exec}(c_k, x_j) \text{ for } k \le n, \qquad Y_{n+1,\,j} \;=\; \hat{y}_j. \tag{3}$$

The per-test agreement between every pair of rows of Y defines the *generalized pass matrix*

$$\mathcal{B} \in \{0,1\}^{(n+1)\times(n+1)\times m}, \qquad \mathcal{B}[k,i,j] \;=\; \mathbf{1}[Y_{k,j} = Y_{i,j}]. \tag{4}$$

$\mathcal{B}$ is the maximal sufficient statistic for execution-agreement reranking, i.e., rerankers whose scores depend on outputs only through pairwise equality (including N$^+$PC and our execution-only baselines); the unstated-code column-slice $\mathcal{B}[1{:}n,\, n{+}1,\, :]$ equals the classical pass matrix B.

2.2 MOTIVATION: FROM TWO DISJOINT SUBSETS TO ALL OF $\mathcal{B}$

Although $\mathcal{B}$ captures every signal an execution-agreement reranker could use, each existing family reads from a *strict subset* of $\mathcal{B}$. The two families differ in which subset they read.

Methods that trust the LLM-generated expected outputs as ground truth. These methods score each candidate by how often its outputs agree with the unstated code's outputs: they read only the $(n{+}1)$-th column of $\mathcal{B}$. With a scorer $\phi : \{0,1\}^{n\times m} \to \mathbb{R}^n$,

$$s \;=\; \phi(B) \;=\; \phi(\mathcal{B}[1:n,\, n{+}1,\, :])\,. \tag{5}$$

Concrete instances of ϕ include Majority Voting (MV), ACES-C, and ACES-O (Sun et al., 2026).

Methods that discard the LLM-generated expected outputs and use cross-code consensus. These methods discard the $(n{+}1)$-th column entirely and read only the $n \times n \times m$ block of pairwise agreements among explicit candidates. Averaging this block over the test axis collapses it to an $n \times n$ consensus matrix, on which a scorer $\psi : \mathbb{R}^{n\times n} \to \mathbb{R}^n$ acts:

$$s \;=\; \psi\left(\frac{1}{m}\sum_{j=1}^{m} \mathcal{B}[1:n,\, 1:n,\, j]\right). \tag{6}$$

Concrete instances differ in the choice of ψ: MBR-exec (Shi et al., 2022) scores each candidate by its row average in the consensus matrix; SRank (To et al., 2024) groups candidates into execution-equivalent clusters and scores via inter-cluster agreement.

Our generalized view. The two prior subsets of $\mathcal{B}$ are *disjoint*: trust uses only the $(n{+}1)$-th column (Eq. 5); discard uses only the inter-candidate block (Eq. 6); no existing method uses both. Our framework applies a single joint scorer to all of $\mathcal{B}$. §3 introduces this scorer (N$^+$PC), whose boundary cases connect the two information regimes (§ 3.3).

3 METHOD: $(N{+}1)$ PROPAGATION CONSISTENCY

The generalized pass matrix $\mathcal{B}$ (Eq. 4) records every pairwise per-test agreement among the $n{+}1$ codes but is not a ranking. N$^+$PC constructs $n{+}1$ agreement views, each providing normalized scores and a graph over the n explicit candidates (§§ 3.1 and 3.2). Joint propagation optimizes one shared score vector, with task-adaptive κ weighting the unstated-code view. Fig. 2 illustrates this pipeline; § 3.3 establishes its endpoint reductions.

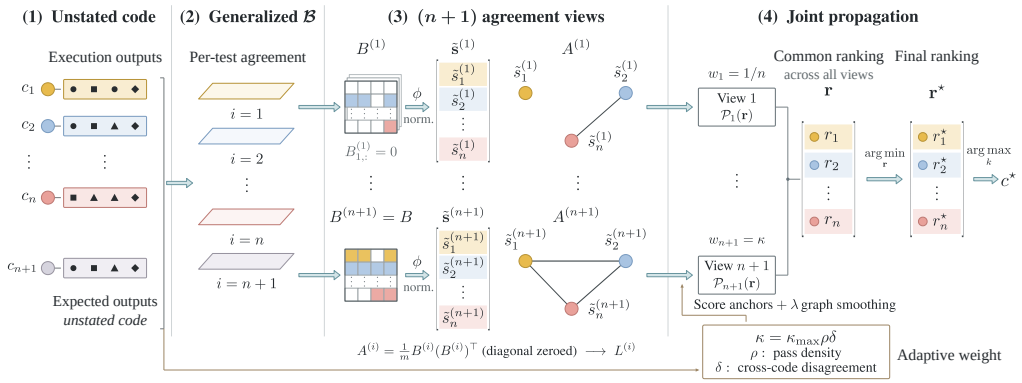


Figure 2: **Overview of N$^+$PC. (1)** The expected outputs trace an unstated code alongside n explicit candidates. **(2)** Pairwise per-test agreements form $\mathcal{B}$. **(3)** The $n{+}1$ views $B^{(i)}$ yield normalized scores $\tilde{\mathbf{s}}^{(i)}$ and graph adjacencies $A^{(i)}$, with Laplacians $L^{(i)}$. "norm." denotes z-score normalization followed by a sigmoid. **(4)** Joint minimization (Eq. 11) yields $\mathbf{r}^\star$, using explicit-view weights $1/n$ and unstated-view weight $\kappa = \kappa_{\max}\rho\delta$. Its largest entry selects an explicit candidate $c^\star$. Colors track candidates; graphs show representative nodes.

3.1 $(N{+}1)$ Agreement Views and Per-View Scores

Agreement view family. Treat each of the $n{+}1$ codes in turn as a hypothesized reference and record its per-test agreement with the n explicit candidates. Zeroing self-comparisons gives one $n \times m$ binary matrix per reference, the *agreement view family*

$$B^{(i)} \in \{0,1\}^{n\times m}, \quad B^{(i)}_{k,j} := \mathcal{B}[k,\, i,\, j] \cdot \mathbf{1}[k \neq i], \quad i \in \{1,\ldots,n{+}1\}. \tag{7}$$

The unstated-code view equals the classical pass matrix, $B^{(n+1)} \equiv B$. The mask $\mathbf{1}[k \neq i]$ is a code-level leave-one-out: on an explicit view $i \leq n$ it zeros the self-row $B^{(i)}_{i,:}$, since code i would otherwise trivially agree with itself and inflate its own view; on the unstated view it has no effect because k ranges only over the n explicit codes.

Per-view scoring. Any pass-matrix reranker $\phi : \{0,1\}^{n\times m} \to \mathbb{R}^n$ from § 2.2 produces a raw score vector $\mathbf{s}^{(i)} := \phi(B^{(i)})$ on each view, which we then z-score and squash through a sigmoid:

$$s^{(i)}_k = \frac{1}{m}\sum_{j=1}^{m} B^{(i)}_{k,j} \ \text{(MV)}, \qquad \tilde{s}^{(i)}_k := \operatorname{sigmoid}\left(\frac{s^{(i)}_k - \mu_i}{\sigma_i}\right). \tag{8}$$

Above, MV is illustrative (ACES-C and ACES-O reweight tests in the same row aggregation); μ_i, σ_i are the empirical mean and std of $\mathbf{s}^{(i)}$, and the normalization puts views on a common scale and bounds outliers so no single view dominates. When $\sigma_i = 0$ we set $\tilde{\mathbf{s}}^{(i)} = \frac{1}{2}\mathbf{1}$ (the view carries no within-view ranking information). N$^+$PC's contribution is *how* it couples the resulting $n{+}1$ normalized scores $\{\tilde{\mathbf{s}}^{(i)}\}$, not the choice of ϕ.

3.2 Coupling $(N{+}1)$ Propagations with Task-Adaptive Weight

Per-view agreement matrix and Laplacian. Each view i induces an agreement structure on pairs of codes: two codes that agree with i on the same tests behave alike *under the hypothesis that i is correct*. We encode this as the view-i agreement matrix

$$A^{(i)} := \frac{1}{m} B^{(i)} \left(B^{(i)}\right)^\top \quad \text{(diagonal zeroed)}, \tag{9}$$

whose entry $A^{(i)}_{k,k'}$ counts the fraction of tests on which codes k, k', and reference i all emit the same output. Its symmetric normalized Laplacian

$$L^{(i)} := I - (D^{(i)})^{-1/2} A^{(i)} (D^{(i)})^{-1/2}, \quad D^{(i)} = \operatorname{diag}(A^{(i)}\mathbf{1}), \tag{10}$$

has a quadratic form $\mathbf{r}^\top L^{(i)}\mathbf{r}$ whose minimization pulls together (degree-normalized) scores of high-$A^{(i)}_{k,k'}$ pairs. For zero-degree nodes (e.g., the self-suppressed row $k = i$ on explicit-code views), we set $(D^{(i)})^{-1/2}_{kk} = 0$, which gives $L^{(i)}_{kk} = 1$ with zero off-diagonal entries on that row; $L^{(i)}$ therefore remains symmetric PSD and the quadratic form is well-defined.

Propagation consistency. Let $\mathbf{r} \in \mathbb{R}^n$ list the explicit candidates' scores in index order. We minimize the weighted sum of the $n{+}1$ view objectives,

$$\mathbf{r}^\star \;=\; \arg\min_{\mathbf{r}\in\mathbb{R}^n} \sum\nolimits_{i=1}^{n+1} w_i\,\mathcal{P}_i(\mathbf{r}), \quad \text{where} \quad \mathcal{P}_i(\mathbf{r}) \;:=\; \|\mathbf{r} - \tilde{\mathbf{s}}^{(i)}\|_2^2 + \lambda\,\mathbf{r}^\top L^{(i)}\,\mathbf{r}, \tag{11}$$

each $\mathcal{P}_i$ is a Tikhonov-style graph-Laplacian regularized objective (Zhu et al., 2003; Zhou et al., 2003; Belkin et al., 2004), whose minimizer $(I + \lambda L^{(i)})^{-1}\,\tilde{\mathbf{s}}^{(i)}$ is one Laplacian propagation; their joint minimization produces a single ranking consistent with all $n{+}1$ propagations. Each of the n explicit-code views has weight $w_i = 1/n$, summing to 1; the unstated-code view's weight $w_{n+1} = \kappa$ is task-adaptive, calibrated against this total.

Task-adaptive weight κ. We use a deterministic calibration rule that raises κ only when two observable proxies both indicate trust in the unstated-code view: the unstated code is supported by the candidates, *and* the cross-code consensus is too fragmented to rely on. We combine them multiplicatively (validated by the ablations in § 4.3):

$$\kappa \;:=\; \kappa_{\max} \cdot \underbrace{\tfrac{1}{nm}\sum\nolimits_{k,j} B^{(n+1)}_{k,j}}_{\rho\text{: pass density on } B^{(n+1)}} \cdot \underbrace{\tfrac{1}{nm}\sum\nolimits_{j} \big|\{Y_{k,j}\}_{k=1}^{n}\big|}_{\delta\text{: cross-code disagreement on } Y_{1:n,:}} . \tag{12}$$

The pass density $\rho \in [0, 1]$ captures the first condition: high ρ is direct evidence that the unstated code is correct, since many independent candidates produce its outputs. The cross-code disagreement $\delta \in [1/n, 1]$ captures the second: high δ means candidates scatter across distinct outputs (one per code at the extreme), so the cross-code consensus is weak. κ is high only when both ρ and δ are high, with $\kappa_{\max}$ as the strict upper bound.

Closed form for $\mathbf{r}^\star$. Substituting the task-adaptive κ (Eq. 12) as w_{n+1} and minimizing the joint objective (Eq. 11) over $\mathbf{r}$ yields the closed-form solution

$$\mathbf{r}^\star \;=\; \Big((\kappa{+}1)\,I \;+\; \frac{\lambda}{n}\sum\nolimits_{i=1}^{n} L^{(i)} + \lambda\kappa\,L^{(n+1)}\Big)^{-1}\Big(\frac{1}{n}\sum\nolimits_{i=1}^{n}\tilde{\mathbf{s}}^{(i)} + \kappa\,\tilde{\mathbf{s}}^{(n+1)}\Big). \tag{13}$$

All views enter one joint optimization of the shared $\mathbf{r}$. We select the explicit candidate $c^\star$ with the largest eligible entry of $\mathbf{r}^\star$ (Algorithm 1).

3.3 Endpoint Reduction

For any inner scorer ϕ, the boundary configurations of (κ, λ) connect the reference-trusting and reference-free information regimes of § 2.2 within one objective.

Proposition 1 (Endpoint reduction; proof in § A.3)**.** *The closed-form $\mathbf{r}^\star$ in Eq. 13 satisfies:*

- ***(i) Reduces to methods that trust the expected outputs.*** *As $\kappa \to \infty$ and $\lambda = 0$, $\mathbf{r}^\star \to \tilde{\mathbf{s}}^{(n+1)}$, which ranks identically to $\phi(B^{(n+1)})$; reductions include $\phi =$ MV, ACES, and CodeT.*
- ***(ii) Reduces to the discard-expected information regime.*** *With $\kappa = 0$ and any finite $\lambda \geq 0$, $\mathbf{r}^\star = (I + \lambda\bar{L})^{-1}\bar{\mathbf{s}}$, where $\bar{L} := \frac{1}{n}\sum_{i=1}^{n} L^{(i)}$ and $\bar{\mathbf{s}} := \frac{1}{n}\sum_{i=1}^{n}\tilde{\mathbf{s}}^{(i)}$ depend only on explicit-candidate agreements, with no expected-output signal.*

The trust-expected limit requires $\lambda = 0$, whereas $\kappa = 0$ removes the expected-output signal at any finite λ. These reductions hold for *any* inner scorer ϕ and connect both information regimes; in the interior, task-adaptive κ (Eq. 12) fuses their evidence in the same optimization framework (§4).

4 Experiments

This section empirically evaluates N$^+$PC: § 4.1 fixes the protocol, § 4.2 reports headline Pass@k, § 4.3 attributes the gains to each component, and § 4.4 traces them across tasks.

Table 1: Pass@k (%) on HumanEval, HumanEval$^+$, and MBPP. Post-hoc methods rerank GPT-3.5-Turbo candidates (n≈200, m≈500). Subscripts denote the change from GPT-3.5-Turbo direct inference; Δ rows give the gain of +Ours over the baseline directly above. **Bold**/underline: best/second-best within each group. Shaded : ours.

	HumanEval			HumanEval$^+$			MBPP		
Method	Pass@1	Pass@2	Pass@5	Pass@1	Pass@2	Pass@5	Pass@1	Pass@2	Pass@5
Direct inference									
GPT-3.5-Turbo	68.38	76.24	83.15	58.75	66.58	73.96	66.80	72.34	76.60
GPT-4	81.48	86.31	90.46	70.52	75.48	79.54	71.26	74.27	76.99
DeepSeek-Coder	79.30	–	–	–	–	–	70.00	–	–
WizardCoder	73.20	–	–	–	–	–	61.20	–	–
CodeLlama	62.20	–	–	–	–	–	62.20	–	–
Post-hoc reranking using only code × test execution									
MBR-exec	72.96 +4.6	76.47 +0.2	79.00 -4.2	62.12 +3.4	67.08 +0.5	71.38 -2.6	70.79 +4.0	73.14 +0.8	**74.85** -1.8
CodeT	78.05 +9.7	78.05 +1.8	78.30 -4.9	67.87 +9.1	68.75 +2.2	69.65 -4.3	71.90 +5.1	71.95 -0.4	72.02 -4.6
Majority Voting	80.49 +12.1	82.93 +6.7	83.54 +0.4	69.51 +10.8	73.17 +6.6	76.83 +2.9	68.62 +1.8	70.49 -1.9	72.83 -3.8
+ N$^+$PC	82.93 +14.6	83.54 +7.3	83.54 +0.4	**76.22** +17.5	76.83 +10.3	77.44 +3.5	**74.00** +7.2	**74.00** +1.7	74.00 -2.6
Δ	↑2.44	↑0.61	0.00	↑6.71	↑3.66	↑0.61	↑5.38	↑3.51	↑1.17
ACES-C	82.93 +14.6	82.93 +6.7	82.93 -0.2	71.34 +12.6	71.95 +5.4	74.39 +0.4	71.19 +4.4	71.43 -0.9	72.37 -4.2
+ N$^+$PC	**84.76** +16.4	84.76 +8.5	84.76 +1.6	75.61 +16.9	76.22 +9.6	76.83 +2.9	**74.00** +7.2	**74.00** +1.7	74.24 -2.4
Δ	↑1.83	↑1.83	↑1.83	↑4.27	↑4.27	↑2.44	↑2.81	↑2.57	↑1.87
ACES-O	84.15 +15.8	85.98 +9.7	**86.59** +3.4	74.39 +15.6	75.61 +9.0	79.88 +5.9	72.37 +5.6	73.54 +1.2	73.77 -2.8
+ N$^+$PC	**84.76** +16.4	**86.59** +10.4	**86.59** +3.4	**76.22** +17.5	**78.66** +12.1	**81.10** +7.1	73.77 +7.0	**74.00** +1.7	74.47 -2.1
Δ	↑0.61	↑0.61	0.00	↑1.83	↑3.05	↑1.22	↑1.40	↑0.46	↑0.70
Post-hoc reranking using additional information									
SC+Spec	73.86 +5.5	73.93 -2.3	74.10 -9.1	63.50 +4.8	64.70 -1.9	65.67 -8.3	71.70 +4.9	71.73 -0.6	71.82 -4.8
Self-collaborate	74.40 +6.0	–	–	–	–	–	68.20 +1.4	–	–
MPSC	74.17 +5.8	77.02 +0.8	78.53 -4.6	65.05 +6.3	69.76 +3.2	71.72 -2.2	69.34 +2.5	70.06 -2.3	71.85 -4.8
$\mathcal{DS}^3$	81.71 +13.3	82.32 +6.1	82.32 -0.8	72.56 +13.8	73.78 +7.2	75.00 +1.0	75.88 +9.1	77.28 +4.9	78.45 +1.9

4.1 Setup

Benchmarks. We evaluate on HumanEval (HE) (Chen et al., 2021) (164 tasks), HumanEval$^+$ (HE$^+$) (Liu et al., 2023) (164 tasks; stricter tests), and MBPP (Austin et al., 2021) (427 tasks), using the GPT-3.5-Turbo candidate pool and LLM-generated tests from Huang et al. (2024) (n≈200 candidates, m≈500 tests per task). We report Pass@k for $k \in \{1, 2, 5\}$.

Baselines and inner scorers. We compare against execution-only rerankers consuming the same traces as N$^+$PC: **Majority Voting** (MV; pass-rate on $B^{(n+1)}$), MBR-exec (Shi et al., 2022) (pairwise output comparison), CodeT (Chen et al., 2023) (consensus-set scoring), and **ACES-C** (closed-form) and **ACES-O** (optimization-based) from Sun et al. (2026). MV, ACES-C, and ACES-O serve as both baselines and N$^+$PC's inner scorers ϕ. For reference, we further include methods using signals beyond execution (SC+Spec, Self-collaborate (Dong et al., 2024), MPSC (Huang et al., 2024)) and direct inference from GPT-3.5-Turbo at top-k.

N$^+$PC introduces only two hyperparameters, λ and $\kappa_{\max}$ (values in § C.5). All rerankers break ties by smallest candidate index. Recipe and protocol are in Appendix B; extended results (ablations, sensitivity, robustness, κ distribution, Pass@k, per-bin matchup) are in Appendix C.

4.2 Main Results

Tab. 1 reports Pass@k across all methods and benchmarks; four patterns emerge.

(i) Zero degradation across all 27 cells. N$^+$PC *never lowers* any inner scorer on any of the 27 (inner × dataset × k) cells: every Δ row in Tab. 1 is non-negative. This empirical zero-degradation is consistent with the endpoint fallback in Proposition 1 (i).

Table 2: Ablations on N$^+$PC (inner: ACES-O). Pass@1 (%); subscripts and Δ give per-dataset and avg drops from the *Full method*. Full Pass@k in § C.4.

Variant	HE	HE$^+$	MBPP	Avg Pass@1	Avg Δ
Full method (N$^+$PC, ACES-O inner)	84.76	76.22	73.77	**78.25**	
Endpoint reductions (Proposition 1)					
Endpoint (i): trust expected outputs†	84.15 $_{-0.61}$	74.39 $_{-1.83}$	72.37 $_{-1.40}$	76.97	$\downarrow$1.28
Endpoint (ii): discard expected outputs‡	75.61 $_{-9.15}$	66.46 $_{-9.76}$	72.60 $_{-1.17}$	71.56	$\downarrow$6.69
Component simplifications					
w/o adaptive κ (fixed tuned κ)	82.93 $_{-1.83}$	74.39 $_{-1.83}$	72.60 $_{-1.17}$	76.64	$\downarrow$1.61
w/o δ factor in κ	83.54 $_{-1.22}$	73.17 $_{-3.05}$	72.83 $_{-0.94}$	76.51	$\downarrow$1.74
w/o ρ factor in κ	82.93 $_{-1.83}$	75.00 $_{-1.22}$	73.07 $_{-0.70}$	77.00	$\downarrow$1.25
w/o per-view $L^{(i)}$ (single L)$^\maltese$	84.15 $_{-0.61}$	74.39 $_{-1.83}$	72.83 $_{-0.94}$	77.12	$\downarrow$1.13
w/o smoothing (λ=0, score fusion only)	84.76 $_{-0.00}$	74.39 $_{-1.83}$	73.07 $_{-0.70}$	77.41	$\downarrow$0.84

† $\kappa\to\infty$, λ=0: reduces to ACES-O. ‡ κ=0 (tuned λ): reduces to cross-code agreement scoring.
$^\maltese$ $\{L^{(i)}\}$ replaced by one shared L from pairwise output-match $S_{k,k'} = \frac{1}{m}\sum_j \mathbf{1}[Y_{k,j}{=}Y_{k',j}]$.

(ii) Simpler inners gain the most; stronger inners do not saturate. The dataset-averaged Pass@1 gain shrinks monotonically with inner strength (MV +4.84, ACES-C +2.97, ACES-O +1.28) but stays strictly positive. The pattern fits the framework's purpose: simpler inners leave more inter-code structure unutilized, so N$^+$PC supplies more there; stronger inners still benefit from the $(n{+}1)$-view aggregation that no single view captures.

(iii) N$^+$PC cross-validates LLM-test agreement against cross-code consensus. HE and HE$^+$ share reranker inputs and differ only in the held-out labels, so each method outputs identical rankings on the two. N$^+$PC's gain over its inner widens under HE$^+$'s stricter labels (MV Pass@1: +2.44 $\rightarrow$ +6.71; ACES-O: +0.61 $\rightarrow$ +1.83): unlike the inner, N$^+$PC requires cross-code execution agreement, filtering out coincidental LLM-test passes that stricter labels expose.

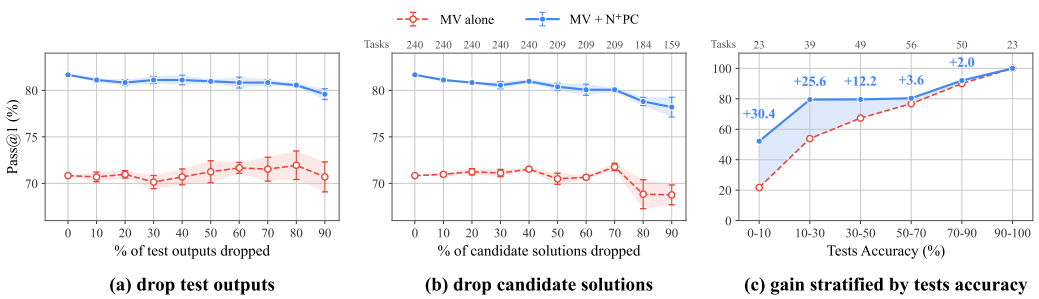
(iv) N$^+$PC matches methods that use extra signals. Group 3 baselines (SC+Spec, MPSC, DS^3) augment execution with extra signals: specifications, multi-agent LLM calls, or static analysis. Using only execution traces, ACES-O + N$^+$PC beats every Group 3 method on HE and HE$^+$ at every k.

4.3 Ablations

We ablate each component of N$^+$PC; Tab. 2 groups results into endpoint reductions (Proposition 1) and component simplifications, with full Pass@k in § C.4.

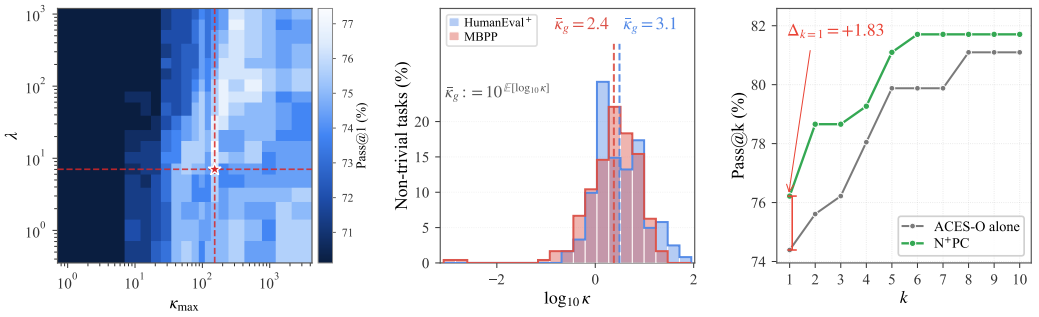
(i) The interior beats either endpoint; the loss shows two distinct asymmetries. (1) Between endpoints: the discard-expected cost dwarfs the trust-expected one. Reducing N$^+$PC to the trust-expected endpoint costs 1.28 avg-Pass@1, while reducing to the discard-expected endpoint costs 6.69. The unstated-code view and the cross-code consensus are complementary signals, and N$^+$PC's adaptive κ *couples* them per task by weighting whichever is more reliable. **(2) Across datasets: the discard-expected cost concentrates on HE/HE$^+$.** The discard-expected cost is $\sim$9.5 Pass@1 on HE/HE$^+$ but only 1.17 on MBPP (Tab. 3), where cross-code consensus alone is near-sufficient. The trust-expected cost, by contrast, is uniform across datasets.

(ii) Each interior component plays a distinct structural role. (1) Adaptive κ: δ outweighs ρ. Dropping δ alone costs 1.74 Pass@1, against 1.25 for ρ. The asymmetry tracks what each factor in Eq. 12 regulates: ρ pushes κ up whenever the LLM expected output has direct candidate support, but cannot detect when cross-code consensus is actually reliable enough to override the expected output; δ supplies this check, pulling κ down when candidates concentrate on a single output. **(2) Per-view $L^{(i)}$: κ steers which Laplacian dominates.** Collapsing the $n{+}1$ Laplacians into one shared L costs 1.13. In the closed form (Eq. 13), κ multiplies $L^{(n+1)}$ but not the cross-code Laplacians $\{L^{(i)}\}_{i\le n}$, so high κ makes the smoothing graph track the unstated-code view and low κ makes it track the cross-code views; collapsing all Laplacians into one removes this differentiation, leaving κ to scale only a fixed graph. **(3) Graph regularization adds over score fusion alone:** 0.84**.** At λ=0



(a) drop test outputs (b) drop candidate solutions (c) gain stratified by tests accuracy

Figure 3: MBPP, Pass@1 on *non-trivial* tasks (both correct and incorrect candidates); N$^+$PC (blue) vs MV alone (red). **(a) Test-output dropout:** fraction of LLM-generated test outputs removed; bands ± 1 s.d. over 3 seeds. **(b) Candidate-solution dropout:** same on candidates; top-axis counts remaining non-trivial tasks. **(c) Stratification by tests accuracy:** tasks binned by the fraction of LLM expected outputs matching a correct candidate's output.



(a) HP sensitivity (b) Adaptive κ distribution (c) Pass@k vs k

Figure 4: N$^+$PC under the ACES-O inner. **(a) Hyperparameter sensitivity:** Pass@1 over the $(\lambda, \kappa_{\max})$ grid on HE$^+$; red crosshair and star mark the $(\lambda, \kappa_{\max})$ values used throughout the paper. **(b) Adaptive κ distribution:** per-task realized κ on HE$^+$ and MBPP; dashed lines mark the geometric means $\bar{\kappa}_g$. **(c) Pass@k vs k on HE$^+$:** ACES-O alone vs $+$N$^+$PC; bracket marks $\Delta_{k=1}$.

the closed form reduces to multi-view score fusion $\mathbf{r} = \frac{1}{\kappa+1}\sum_i w_i \tilde{\mathbf{s}}^{(i)}$, a deterministic weighted average of the $(n+1)$ per-view scores with no graph smoothing. The 0.84 avg-Pass@1 gap between this score-fusion variant and the full method isolates what graph regularization contributes beyond ensembling, by pulling functionally similar candidates together in the closed-form solution.

4.4 Diagnostic Analyses

Figs. 3 and 4 diagnose where gains survive and concentrate, plus how HP, κ, and Pass@k behave.

(i) N$^+$PC keeps its lead under heavy dropout, with lower variance than MV. Fig. 3a,b probe N$^+$PC's sensitivity to input size: panels (a, b) drop random fractions from 0% to 90% of the LLM-generated test outputs and the candidate codes, respectively; bands show ± 1 std over 3 dropout seeds. N$^+$PC's Pass@1 lead over MV is 10.8% on the full set, and only narrows to 8.9% at 90% test-removal and 9.4% at 90% code-removal (ACES-O parallels in § C.3). Across removal levels, N$^+$PC's seed variance is generally below MV's. This rules out the concern that execution-based reranking needs a dense (code $\times$ test) grid: N$^+$PC's inter-code aggregation survives sparse regimes, and adaptive κ dampens MV's seed noise as data thins.

(ii) Gains concentrate exactly where the motivation predicts. Fig. 3c bins tasks by their tests accuracy (the fraction of LLM-generated test outputs matching a ground-truth-correct code's output) and plots the per-bin Pass@1 gain over MV. The gain decreases monotonically: N$^+$PC adds 30.4 points in the leftmost bin (tests accuracy below 10%) and 0 in the rightmost (above 90%), matching the motivation that the LLM expected output is trustworthy only when the unstated code is itself

plausible; N$^+$PC's adaptive κ downweights it precisely when that condition fails, concentrating Tab. 1's aggregate gains on this hard regime.

(iii) N$^+$PC is largely insensitive to $(\lambda, \kappa_{\max})$. Pass@1 is largely flat across a broad region of $(\lambda, \kappa_{\max})$ (Fig. 4a; HE/HE$^+$/MBPP heatmaps in § C.5), dipping only at very small $\kappa_{\max}$ where N$^+$PC reduces toward cross-code consensus alone; the reported $(\lambda, \kappa_{\max})$ sits inside this flat region and thus serves as a robust default.

(iv) κ varies widely across tasks. Fig. 4b plots the per-task realized κ on each dataset; the distribution is broad on both, with geometric means $\bar{\kappa}_g = 3.1$ on HE$^+$ and 2.4 on MBPP (the larger value reflects more reliable LLM expected outputs there), driving the regime dependence in Fig. 3c.

(v) Gains at small k improve selection precision. On HE$^+$, N$^+$PC improves over ACES-O by 1.83 points at k=1 and 3.05 at k=2 (Tab. 1), showing clear gains under small selection budgets. This illustrates N$^+$PC's value as a *precision sharpener* under tight top-k budgets at deployment (per-dataset Pass@k in § C.2).

5 Related Work

Methods that trust the expected outputs. A first family treats the LLM expected output as ground truth and ranks candidates by pooling the pass matrix $B^{(n+1)}$. Members differ in how they pool $B^{(n+1)}$: plain Majority Voting; CodeT (Chen et al., 2023), which groups solutions by pass pattern and weights each group by passing-test count and size; ACES (Sun et al., 2026), which uses leave-one-out AUC weights; or B^4 (Chen et al., 2024a), which approximates the Bayesian-optimal rule under plausible tests. A recent variant only changes the source: ALGO (Zhang et al., 2023a) replaces $\hat{y}_j$ with an LLM-synthesized oracle's outputs, used both to guide generation and to rank candidates post-hoc. All rank against a single expected-output reference, so a wrong reference collapses every member; their pass-matrix scoring corresponds to the $\kappa \to \infty$, $\lambda = 0$ endpoint of N$^+$PC under the appropriate inner ϕ.

Methods that discard the expected outputs. A second family ignores $\hat{y}_j$ and ranks codes by cross-code output agreement: pairwise minimum-Bayes-risk (Shi et al., 2022), cluster size on behavioral equivalence (Li et al., 2022), or an inter-cluster functional-overlap matrix (To et al., 2024). Recent variants keep cross-code agreement as the core but layer extra signals: S* (Li et al., 2025a) clusters candidates by outputs and uses an LLM to disambiguate cluster pairs; DS^3 (Liu et al., 2026) combines a candidate-vs-candidate output-agreement matrix with AST-based static-analysis filtering. These methods exclude the expected-output signal; the κ=0 configuration of N$^+$PC shares this information regime through reference-free graph aggregation, connecting both stances within one objective (Proposition 1).

Other related directions. A first group acts upstream of selection and is complementary to N$^+$PC: CURE (Wang et al., 2025) co-trains code and test generators with reinforcement learning; CoCoEvo (Li et al., 2025b) co-evolves them at inference time; Taherkhani et al. (2026) improves LLM-test quality via self-consistency; Self-debugging (Chen et al., 2024b) refines each candidate rather than ranking them; and Yang et al. (2025) study LLM-test fault coverage. A second group uses signals beyond ours: MPSC (Huang et al., 2024) adds executable specifications; supervised verifiers (Ni et al., 2023; Sun et al., 2024) step outside our training-free regime; and LM-likelihood reranking (Zhang et al., 2023b) bypasses execution entirely.

6 Conclusion

We presented N$^+$PC, built on the insight that *the tests' expected outputs reveal an unstated $(n+1)$-th candidate code on equal footing with the n explicit ones*. N$^+$PC couples $n+1$ Laplacian propagations via task-adaptive κ, with boundary cases (Proposition 1) connecting reference-trusting scoring and reference-free graph aggregation within one objective. As a plug-and-play wrapper around any pass-matrix reranker, N$^+$PC improves every inner baseline and advances the state-of-the-art Pass@k among execution-only rerankers. At its core, N$^+$PC embodies a general perspective: an LLM's candidates and its own evaluation signal need not be a trust-or-discard binary but can be unified within one principled framework. We hope this $(n+1)$-view reframe invites broader work on self-evaluation, including LLM-as-judge pipelines.

AI USE STATEMENT

We used AI tools to aid and polish the writing of this manuscript, including grammar correction and improvements to clarity and fluency. The authors developed the core method, conducted the experiments, and take full responsibility for the scientific content and final manuscript.

REFERENCES

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *Arxiv Preprint Arxiv:2108.07732*, 2021.

Mikhail Belkin, Irina Matveeva, and Partha Niyogi. Regularization and semi-supervised learning on large graphs. In *International conference on computational learning theory*, pp. 624–638. Springer, 2004.

Bei Chen, Fengji Zhang, Anh Nguyen, Daoguang Zan, Zeqi Lin, Jian-Guang Lou, and Weizhu Chen. CodeT: Code generation with generated tests. In *The Eleventh International Conference on Learning Representations*, 2023.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *Arxiv Preprint Arxiv:2107.03374*, 2021.

Mouxiang Chen, Zhongxin Liu, He Tao, Yusu Hong, David Lo, Xin Xia, and Jianling Sun. B^4: Towards optimal assessment of plausible code solutions with plausible tests. In *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*, pp. 1693–1705, 2024a.

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. Teaching large language models to self-debug. In *The Twelfth International Conference on Learning Representations*, 2024b.

Yihong Dong, Xue Jiang, Zhi Jin, and Ge Li. Self-collaboration code generation via ChatGPT. *Acm Transactions on Software Engineering and Methodology*, 33(7), 2024.

Baizhou Huang, Shuai Lu, Xiaojun Wan, and Nan Duan. Enhancing large language models in coding through multi-perspective self-consistency. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (volume 1: Long Papers)*, pp. 1429–1450. Association for Computational Linguistics, 2024.

Dacheng Li, Shiyi Cao, Chengkun Cao, Xiuyu Li, Shangyin Tan, Kurt Keutzer, Jiarong Xing, Joseph E. Gonzalez, and Ion Stoica. S*: Test time scaling for code generation. In *Findings of the Association for Computational Linguistics: Emnlp 2025*, pp. 15964–15978. Association for Computational Linguistics, 2025a.

Kefan Li, Yuan Yuan, Hongyue Yu, Tingyu Guo, and Shijie Cao. CoCoEvo: Co-evolution of programs and test cases to enhance code generation. *Ieee Transactions on Evolutionary Computation*, pp. 1–1, 2025b.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097, 2022.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and LINGMING ZHANG. Is your code generated by chatGPT really correct? rigorous evaluation of large language models for code generation. In *Thirty-Seventh Conference on Neural Information Processing Systems*, 2023.

Ren-Biao Liu, Jiang-Tian Xue, Chao-Zeng Ma, Hui Sun, Xin-Ye Li, and Ming Li. Dynamic-Static synergistic selection method for candidate code solutions with generated test cases. In *Proceedings of the Aaai Conference on Artificial Intelligence*, pp. 32096–32104, 2026.

Ansong Ni, Srini Iyer, Dragomir Radev, Veselin Stoyanov, Wen-tau Yih, Sida Wang, and Xi Victoria Lin. LEVER: Learning to verify language-to-code generation with execution. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 26106–26128. PMLR, 2023.

Freda Shi, Daniel Fried, Marjan Ghazvininejad, Luke Zettlemoyer, and Sida I. Wang. Natural language to code translation with execution. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 3533–3546. Association for Computational Linguistics, 2022.

Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth International Conference on Learning Representations*, 2025.

Hui Sun, Yun-Ji Zhang, Zheng Xie, Ren-Biao Liu, Yali Du, Xin-Ye Li, and Ming Li. Aces: Who tests the tests? leave-one-out auc consistency for code generation. *arXiv preprint arXiv:2604.03922*, 2026.

Zhihong Sun, Yao Wan, Jia Li, Hongyu Zhang, Zhi Jin, Ge Li, and Chen Lyu. Sifting through the chaff: On utilizing execution feedback for ranking the generated code candidates. In *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*, pp. 229–241, 2024.

Hamed Taherkhani, Alireza DaghighFarsoodeh, Mohammad Chowdhury, Hung Viet Pham, and Hadi Hemmati. Consistency meets verification: Enhancing test generation quality in large language models without ground-truth solutions. *Arxiv Preprint Arxiv:2602.10522*, 2026.

Hung Quoc To, Minh Huynh Nguyen, and Nghi D. Q. Bui. Functional overlap reranking for neural code generation. In *Findings of the Association for Computational Linguistics: Acl 2024*, pp. 3686–3704. Association for Computational Linguistics, 2024.

Yinjie Wang, Ling Yang, Ye Tian, Ke Shen, and Mengdi Wang. CURE: Co-evolving coders and unit testers via reinforcement learning. In *The Thirty-Ninth Annual Conference on Neural Information Processing Systems*, 2025.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. Inference scaling laws: An empirical analysis of compute-optimal inference for LLM problem-solving. In *The Thirteenth International Conference on Learning Representations*, 2025.

Zheyuan Yang, Zexi Kuang, Xue Xia, and Yilun Zhao. Can LLMs generate high-quality test cases for algorithm problems? TestCase-eval: A systematic evaluation of fault coverage and exposure. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (volume 2: Short Papers)*, pp. 1050–1063. Association for Computational Linguistics, 2025.

Kexun Zhang, Danqing Wang, Jingtao Xia, William Yang Wang, and Lei Li. ALGO: Synthesizing algorithmic programs with generated oracle verifiers. In *Thirty-Seventh Conference on Neural Information Processing Systems*, 2023a.

Tianyi Zhang, Tao Yu, Tatsunori Hashimoto, Mike Lewis, Wen-tau Yih, Daniel Fried, and Sida Wang. Coder reviewer reranking for code generation. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 41832–41846. PMLR, 2023b.

Dengyong Zhou, Olivier Bousquet, Thomas Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. *Advances in neural information processing systems*, 16, 2003.

Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML)*, pp. 912–919, 2003.

Appendix

Contents

A Method Details **13**

A.1 Output Comparison 13

A.2 Algorithm Details 13

A.3 Endpoint Reduction: Proof of Proposition 1 13

B Experimental Protocol **15**

B.1 Benchmarks and Candidate Pool 15

B.2 Evaluation Metric 15

B.3 Reproducibility 15

B.4 Inner Scorer Implementations 15

B.5 All Tasks vs. Non-Trivial Tasks 16

C Additional Experimental Results **17**

C.1 Per-Bin Comparison Against the Regime-Conditional Best 17

C.2 Pass@k Curves 18

C.3 Robustness Under Pool Dropout (ACES-O Inner) 18

C.4 Pass@k Ablation Breakdown 19

C.5 Hyperparameter Sensitivity 20

C.6 Per-Task Runtime 21

A METHOD DETAILS

§ A.1 defines the output matrix Y used by the algorithm. § A.2 states N$^+$PC end-to-end as Algorithm 1. § A.3 proves Proposition 1: the boundary cases of (κ, λ) yield reference-trusting scoring and reference-free graph aggregation.

A.1 OUTPUT COMPARISON

For each cell, $Y_{k,j} = \mathrm{exec}(c_k, x_j)$ is the value returned by running code c_k on the inputs of test x_j. Outputs are compared across cells under two rules:

- **Successful cells:** cells (k, j) and (i, j) agree iff $Y_{k,j}$ and $Y_{i,j}$ have identical string representations under the canonical `repr` the candidate pool of Huang et al. (2024) provides.
- **Failed cells** (compile error, runtime exception, timeout, or unparseable test): each failed cell receives its own unique tag, so two failures never count as agreement (not even two failures of the same exception type).

The same equivalence and failure policy apply to all baselines in this paper; we do not re-canonicalize outputs separately for N$^+$PC. The agreement view tensor records these equalities, $\mathcal{B}[k, i, j] = \mathbf{1}[\text{cells } (k, j) \text{ and } (i, j) \text{ agree}]$; the pass matrix is its slice against the LLM expected-output column, $B^{(n+1)}_{k,j} = \mathcal{B}[k, n{+}1, j]$.

A.2 ALGORITHM DETAILS

Algorithm 1 states N$^+$PC on a single task in four phases that mirror §3: phase (I) builds the $(n{+}1)$ agreement views and per-view scores (§ 3.1); phase (II) computes the task-adaptive scalar κ and the view weights w_i (Eq. 12); phase (III) builds the per-view Laplacians and their weighted sum L_w (Eqs. (9) and (10)); phase (IV) solves the closed-form ranking (Eq. 13) via a single eigendecomposition. The inputs are the generalized pass tensor $\mathcal{B}$ (Eq. 4), the output matrix $Y \in \mathcal{O}^{(n+1)\times m}$ (§ A.1) extended by the unstated-code row $Y_{n+1,:} = \hat{y}$, an inner scorer $\phi : \{0,1\}^{n\times m} \to \mathbb{R}^n$, and the hyperparameters $\lambda, \kappa_{\max}$. The output is a per-code score vector $\mathbf{r} \in \mathbb{R}^n$, from which Pass@$k$ is read by selecting its top-k entries.

Closed-form solve via eigendecomposition. Phase (IV) inverts $(\kappa{+}1)I + \lambda L_w$ via a single eigendecomposition of the symmetric $n \times n$ matrix L_w rather than forming the full inverse: writing $L_w = U\Lambda U^\top$, the inverse acts elementwise in the eigenbasis as $1/\big((\kappa{+}1)\mathbf{1} + \lambda\,\mathrm{diag}(\Lambda)\big)$. The remaining work (rhs projection, elementwise scale, back-projection) is $O(n^2)$ and dwarfed by the $O(n^3)$ eigendecomposition.

When stacking N$^+$PC on top of a reranker that already supplies candidate scores (e.g., DS^3 (Liu et al., 2026)), we substitute that score as the per-view input $\tilde{\mathbf{s}}^{(i)}$ in Algorithm 1 and keep phases (II)–(IV) unchanged; N$^+$PC then graph-regularizes the substituted score over the inter-code agreement structure of the same candidate pool. Empirically, this raises HE/HE$^+$ Pass@k on top of DS^3 by 4–7 points, while on MBPP the candidate pool is small enough that the propagation contributes a near-zero Δ.

A.3 ENDPOINT REDUCTION: PROOF OF PROPOSITION 1

We prove each endpoint by direct substitution into the closed form (Eq. 13).

Endpoint (i): trust the expected outputs. Fix any inner ϕ and assume $\rho > 0$, meaning at least one candidate passes at least one test. Since $\delta \geq 1/n$, letting $\kappa_{\max} \to \infty$ gives $\kappa = \kappa_{\max}\rho\delta \to \infty$. Setting $\lambda = 0$ in Eq. 13 zeroes the Laplacian terms, so the closed form reduces to a simple weighted average of the per-view scores:

$$\mathbf{r}^\star \;=\; \frac{1}{\kappa+1}\left(\kappa\,\tilde{\mathbf{s}}^{(n+1)} + \frac{1}{n}\sum_{i=1}^{n}\tilde{\mathbf{s}}^{(i)}\right).$$

As $\kappa \to \infty$, the $\kappa\,\tilde{\mathbf{s}}^{(n+1)}$ term dominates the numerator and dividing by $\kappa{+}1$ yields

$$\lim_{\kappa\to\infty}\mathbf{r}^\star \;=\; \tilde{\mathbf{s}}^{(n+1)},$$

Algorithm 1 N$^+$PC (per-task inference)

Require: Pass tensor $\mathcal{B}$, outputs Y, inner ϕ, hyperparameters $\lambda, \kappa_{\max}$.

// (I) Build $(n{+}1)$ views and per-view scores (Eqs. (7) and (8))

for $i = 1, \ldots, n{+}1$ **do**
  $B^{(i)}_{k,j} \leftarrow \mathcal{B}[k,\, i,\, j] \cdot \mathbf{1}[k \neq i]$
  $\mathbf{s}^{(i)} \leftarrow \phi(B^{(i)})$; $\tilde{s}^{(i)}_k \leftarrow \mathrm{sigmoid}\big((s^{(i)}_k - \mu_i)/\sigma_i\big)$
end for

// (II) Task-adaptive weight (Eq. 12)

$\rho \leftarrow \frac{1}{nm}\sum_{k,j} B^{(n+1)}_{k,j}$; $\delta \leftarrow \frac{1}{nm}\sum_j |\{Y_{k,j}\}_{k=1}^n|$; $\kappa \leftarrow \kappa_{\max} \cdot \rho \cdot \delta$
View weights: $w_i = 1/n$ for $i \leq n$; $w_{n+1} = \kappa$

// (III) Per-view Laplacians and weighted sum (Eqs. (9) and (10))

for $i = 1, \ldots, n{+}1$ **do**
  $A^{(i)} \leftarrow \frac{1}{m} B^{(i)} (B^{(i)})^\top$ (diagonal zeroed); $D^{(i)} = \mathrm{diag}(A^{(i)}\mathbf{1})$
  $L^{(i)} \leftarrow I - (D^{(i)})^{-1/2}\, A^{(i)}\, (D^{(i)})^{-1/2}$
end for
$L_w \leftarrow \kappa\, L^{(n+1)} + \frac{1}{n}\sum_{i=1}^n L^{(i)}$

// (IV) Closed-form solve (Eq. 13)

Form the right-hand side $\mathbf{b} \leftarrow \sum_{i=1}^{n+1} w_i\, \tilde{\mathbf{s}}^{(i)}$, eigendecompose $L_w = U \Lambda U^\top$, then

$$\mathbf{r} \;\leftarrow\; U\left(\frac{U^\top \mathbf{b}}{(\kappa{+}1)\mathbf{1} + \lambda\, \mathrm{diag}(\Lambda)}\right),$$

where the division is elementwise.
Mask $\mathbf{r}_k \leftarrow -\infty$ for codes with no finite view score.
return $\mathbf{r}$.

the normalized scores on the unstated-code view alone. Per-view z-score followed by sigmoid preserves the ordering of nonconstant scores; constant scores remain tied under Eq. 8. Thus the limiting score vector ranks identically to $\phi(B^{(n+1)})$, the inner-scorer output on the LLM expected-output pass matrix. Choosing $\phi = \mathrm{MV}$ reduces to Majority Voting; $\phi = \mathrm{ACES{-}C}$ or $\mathrm{ACES{-}O}$ reduces to ACES (Sun et al., 2026); ϕ as the consensus-set counter reduces to CodeT (Chen et al., 2023).

Endpoint (ii): discard the expected outputs. Set $\kappa = 0$ and fix any finite $\lambda \geq 0$. With $\kappa = 0$, the unstated-code view weight $w_{n+1} = \kappa$ drops to zero and Eq. 13 reduces to

$$\mathbf{r}^\star \;=\; \big(I + \lambda \bar{L}\big)^{-1} \bar{\mathbf{s}}, \quad \text{with} \quad \bar{\mathbf{s}} := \frac{1}{n}\sum_{i=1}^n \tilde{\mathbf{s}}^{(i)}, \quad \bar{L} := \frac{1}{n}\sum_{i=1}^n L^{(i)}.$$

This $\mathbf{r}^\star$ is equivalently the minimizer of $\|\mathbf{r} - \bar{\mathbf{s}}\|_2^2 + \lambda\, \mathbf{r}^\top \bar{L}\, \mathbf{r}$, a Tikhonov-regularized fit of $\bar{\mathbf{s}}$ with smoothness penalty $\bar{L}$. Every explicit-code view $B^{(i)}$, $i \leq n$, depends only on agreements among the explicit candidates, so neither $\bar{\mathbf{s}}$ nor $\bar{L}$ uses the LLM-generated expected outputs. Because $\bar{L}$ is positive semidefinite, $I + \lambda \bar{L}$ is positive definite and the solution is unique. This establishes reference-free graph aggregation within the information regime of methods that discard the expected outputs, without requiring a limit in λ.

B EXPERIMENTAL PROTOCOL

This part documents the benchmarks and candidate pool we evaluate on (§ B.1), the evaluation metric (§ B.2), the reproducibility setup (§ B.3), inner-scorer implementation choices (§ B.4), and how analyses split between all and non-trivial tasks (§ B.5). Additional experimental results are in Appendix C.

B.1 BENCHMARKS AND CANDIDATE POOL

Benchmarks. We evaluate on three Python function-completion benchmarks:

- **HumanEval (HE)** (Chen et al., 2021): 164 hand-written programming problems, each with a natural-language description, function signature, and a set of unit-test assertions.
- **HumanEval$^+$ (HE$^+$)** (Liu et al., 2023): the same 164 HE tasks with substantially stricter test suites from EvalPlus, exposing more incorrect candidates than HE alone.
- **MBPP** (Austin et al., 2021): 427 crowd-sourced Python programming problems with a similar function-based structure but a distinct problem distribution.

Candidate pool and tests. For each task we use the GPT-3.5-Turbo candidate pool and LLM-generated tests released by Huang et al. (2024): roughly $n \approx 200$ candidate codes and $m \approx 500$ LLM-generated tests per task. The candidate pool is the only source of randomness upstream of the reranker; we use it verbatim and do not regenerate it.

B.2 EVALUATION METRIC

We report Pass@k for $k \in \{1, 2, 5\}$, with denominators $164/164/427$ for HE/HE$^+$/MBPP. Per-task Pass@k is computed differently for the two method groups, both averaged over tasks t:

- **Direct (no reranker).** Without a reranker, no fixed top-k exists; following Chen et al. (2021) we use the unbiased Pass@k estimator,

$$\text{Pass@}k \;\coloneqq\; \mathbb{E}_t\left[\,1 \;-\; \frac{\binom{n_t - c_t}{k}}{\binom{n_t}{k}}\,\right],$$

  where n_t is the number of sampled candidates for task t and c_t the count among them that pass the benchmark's hidden test suite. This equals the expected probability that a uniformly random k-subset of n_t candidates contains at least one passing one.
- **Rerankers** (including N$^+$PC). The reranker produces a deterministic top-k set $\mathcal{T}_k(t)$, and Pass@k is

$$\text{Pass@}k \;\coloneqq\; \mathbb{E}_t[\,\mathbf{1}[\,\exists\, i \in \mathcal{T}_k(t) \;:\; c_{t,i} = 1\,]\,],$$

  where $c_{t,i} = 1$ iff candidate i of task t passes the hidden suite; no combinatorial averaging is needed.

B.3 REPRODUCIBILITY

N$^+$PC's scoring is deterministic given the candidate pool, so every number in Tabs. 1 to 3 reproduces exactly from one run. The robustness analysis in § 4.4 (i), which introduces randomness via dropout, reports ± 1 std over three dropout seeds in its figures.

Hardware and software. All experiments run on a single NVIDIA A100 (40 GB, PCIe) on a server with AMD EPYC 7H12 64-core CPUs and 1 TiB RAM, under Ubuntu 20.04, Python 3.12, and PyTorch 2.4.1 with CUDA 12.1.

B.4 INNER SCORER IMPLEMENTATIONS

N$^+$PC wraps an inner scorer ϕ (§ 4.1) that maps the pass matrix $B \in \{0, 1\}^{n \times m}$ to a per-code score vector. We use three.

Majority Voting (MV). A code is scored by its mean pass-rate over the m tests,

$$s_i^{\mathrm{MV}} \;:=\; \frac{1}{m}\sum_{j=1}^{m} B_{i,j}.$$

Every test contributes equally; no parameters.

ACES-C and ACES-O (Sun et al., 2026). Both weight tests non-uniformly: a code's score has the form $s_i^{\mathrm{ACES}} = \sum_j w_j B_{i,j}$ for test weights w_j derived from each test's discriminative power. ACES-C computes w_j in closed form from a per-test AUC-style statistic; ACES-O learns w_j by Adam optimization of a margin-style objective. We re-implement both directly from the ACES paper and follow its hyperparameter settings without further tuning.

B.5 All Tasks vs. Non-Trivial Tasks

A task is *non-trivial* if its candidate pool contains at least one candidate that passes and at least one that fails the benchmark's hidden test suite. A trivial task (all candidates pass, or all candidates fail) admits the same Pass@k regardless of the reranker, so it carries no signal about reranking quality; we still count it in the overall Pass@k denominator but exclude it from per-task analyses whose signal lies in how the reranker orders the mixed pool. Specifically:

- **Pass@k on all tasks** (denominators $164/164/427$): the main results (Tab. 1) and ablations (Tabs. 2 and 3) report on all tasks to match the evaluation setup of prior baselines for direct comparison; the hyperparameter sensitivity heatmaps (Fig. 4a, Fig. 8), Pass@k vs k curves (Fig. 4c, Fig. 6), and the per-bin matchup (Fig. 5) likewise report on all tasks, since they characterize absolute Pass@k or its per-bin breakdown.
- **Non-trivial tasks only:** the robustness analysis under pool dropout (Fig. 3a, b and Fig. 7a, b), the tests-accuracy stratification (Fig. 3c and Fig. 7c), and the realized κ distribution (Fig. 4b).

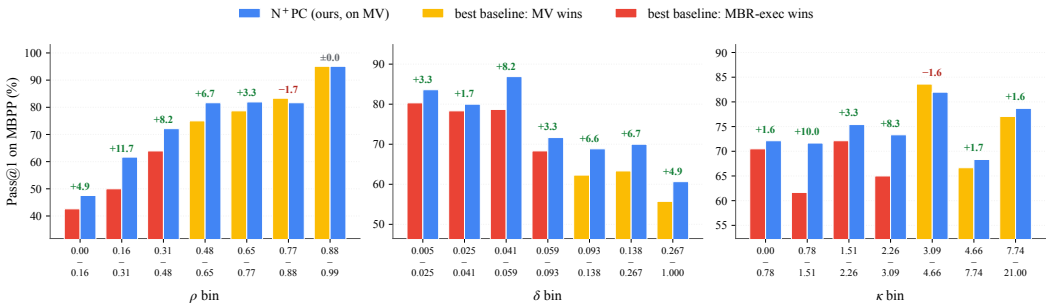


Figure 5: Per-bin Pass@1 on MBPP, with tasks split into 7 equal-frequency bins (~ 60 tasks per bin) along each of the three axes defined in Eq. 12. In each bin we compare N$^+$PC on top of MV (blue) against the stronger of {MV, MBR-exec}, colored by the winning baseline (yellow = MV, red = MBR-exec); the number above each pair is $\Delta = N^+PC -$ best baseline (green for gain, red for loss). **(a) Pass density ρ:** mean pass-rate of candidates on the LLM-generated tests. **(b) Cross-code disagreement δ:** mean distinct-output count per test input, divided by n. **(c) Adaptive view weight κ:** N$^+$PC's task-adaptive weight on the unstated-code view.

C Additional Experimental Results

This part extends the empirical results of §4: a per-bin matchup of N$^+$PC against the regime-conditional best baseline (§ C.1); Pass@k curves on all three benchmarks (§ C.2); pool-dropout robustness under the ACES-O inner (§ C.3); the full Pass@k breakdown of the ablations (§ C.4); hyperparameter sensitivity across datasets (§ C.5); and per-task runtime (§ C.6).

C.1 Per-Bin Comparison Against the Regime-Conditional Best

§ 4.4 reports N$^+$PC's behavior relative to MV alone (Fig. 3) and to its own task-adaptive mechanism (Fig. 4). Here we compare N$^+$PC on MBPP against two baselines representing the information regimes connected by Proposition 1: MV (trust-expected) and MBR-exec (discard-expected) (Shi et al., 2022). In each bin we take the stronger of the two as the envelope.

(i) MV and MBR-exec cover complementary regimes. On both the ρ and δ panels, the winning baseline flips at a single crossover: MBR-exec dominates the low-ρ, low-δ regime, and MV dominates the high-ρ, high-δ regime. This reflects each baseline's scoring rule. MV scores candidates by their pass rate over the LLM-generated tests (the trust-expected family from Proposition 1), which works well when high ρ and δ make per-test agreement informative. MBR-exec instead ignores the expected outputs entirely and scores by cross-candidate output-agreement on the raw outputs (the discard-expected family), which works better when low ρ and δ make the expected outputs unreliable. Neither baseline alone covers both regimes.

(ii) Panel-wide trends match task difficulty. On the ρ panel, all bars rise broadly with ρ, from around 40% at the lowest bin to near 95% at the highest. Higher ρ corresponds to candidate pools where many codes pass the LLM-generated tests; such pools tend to be easier overall, and every reranker achieves higher Pass@1. On the δ panel, the opposite holds: all bars trend downward as δ grows, from around 80% at the lowest bin to around 55% at the highest. Higher δ corresponds to pools where candidates produce widely diverging outputs; these are harder tasks for any reranker, and Pass@1 drops accordingly. These trends move N$^+$PC and both baselines together, so they reflect task difficulty rather than method-specific behavior.

(iii) N$^+$PC tracks and usually beats the regime-conditional envelope. N$^+$PC on top of MV matches or exceeds the envelope in almost every bin, and strictly improves over it in most. The matching is structural rather than coincidental: N$^+$PC's adaptive κ (Eq. 12) uses the same ρ and δ that drive the baseline crossover. When ρ and δ are high (MV's regime), a higher κ increases the relative weight of the unstated-code view. When ρ and δ are low (MBR-exec's regime), κ drops, shifting weight toward the cross-code views; at $\kappa = 0$, only these views contribute (Proposition 1 (ii)).

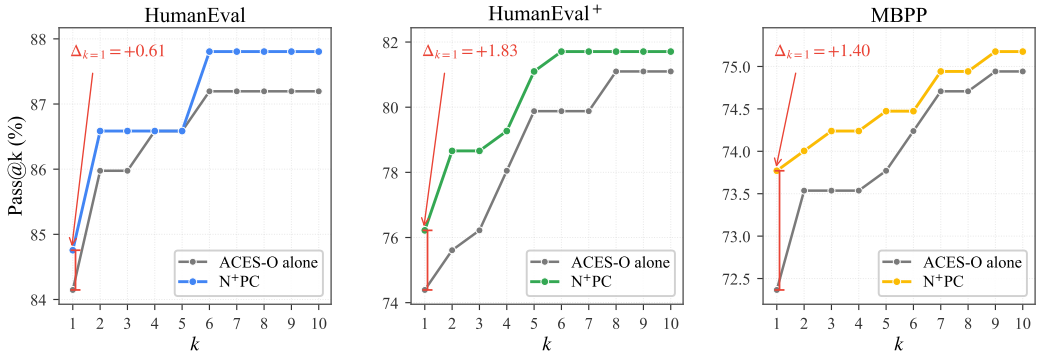
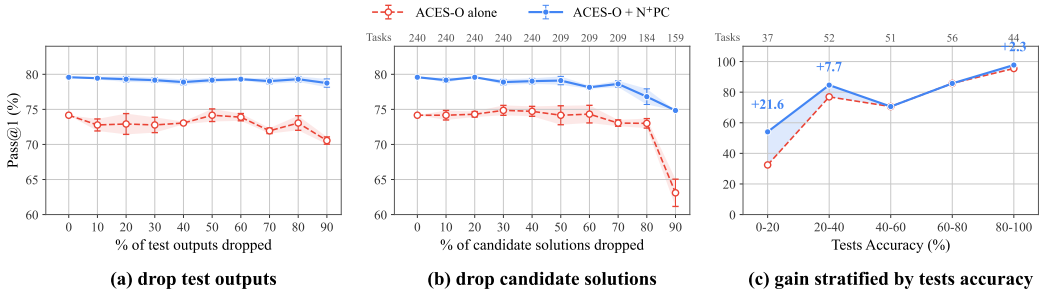


Figure 6: Pass@k vs k, ACES-O alone versus ACES-O+N$^+$PC on each benchmark. Red annotations mark $\Delta_{k=1} = \text{Pass@1}(N^+PC) - \text{Pass@1}(\text{ACES}-\text{O alone})$.



(a) drop test outputs (b) drop candidate solutions (c) gain stratified by tests accuracy

Figure 7: MBPP, Pass@1 on *non-trivial* tasks that have both correct and incorrect candidates; N$^+$PC (blue) vs ACES-O alone (red). **(a) Test-output dropout:** varying fractions of LLM-generated test outputs removed; bands are ± 1 standard deviation over 3 seeds. **(b) Candidate-solution dropout:** same on candidates; gray top-axis numbers count non-trivial tasks remaining. **(c) Stratification by tests accuracy:** tasks binned by ground-truth tests accuracy, the fraction of LLM expected outputs matching a correct candidate's output.

Both pivots happen automatically per task, so N$^+$PC not only follows whichever baseline is locally stronger but adds further gain on top of it in most bins.

C.2 Pass@k Curves

§ 4.4 (v) reports Pass@k on HE$^+$ for ACES-O versus ACES-O+N$^+$PC; Fig. 6 extends to all three benchmarks for $k \in \{1, \ldots, 10\}$.

(i) Zero drop at every k on every dataset. The N$^+$PC curve sits on or above ACES-O alone for all $k \in \{1, \ldots, 10\}$ on all three benchmarks, extending the no-drop pattern of the main results (Tab. 1, which reports only $k \in \{1, 2, 5\}$) to a continuous k sweep.

(ii) Gains under small selection budgets. The main results show gains under tight selection budgets: on HE$^+$, the improvement is 1.83 points at k=1 and 3.05 at k=2 (Tab. 1). These gains demonstrate the benefit of improving the top of the ranking when only a few candidates can be selected.

C.3 Robustness Under Pool Dropout (ACES-O Inner)

§ 4.4 (i) and (ii) study robustness on MBPP with the MV inner; here we run the same protocol with the ACES-O inner, and the main-text conclusions carry over.

(i) Heavy dropout neither erodes N$^+$PC's lead nor inflates its seed variance. Fig. 7a,b zero out from 0% to 90% of LLM-generated test outputs and of candidate codes respectively (stratified

Table 3: Full ablation table with the complete Pass@1/2/5 breakdown. Inner: ACES-O (Split recipe). Mirrors Tab. 2 row-for-row in the first two groups and adds two further single-L code-similarity variants (Jaccard, cosine on $B^{(n+1)}$). Red subscripts on the rightmost column show the drop in avg-Pass@1 from the full method.

Variant	**HumanEval** P@1/2/5	**HumanEval$^+$** P@1/2/5	**MBPP** P@1/2/5	Avg P@1
Full method (N$^+$PC, ACES-O inner)	84.76/86.59/86.59	76.22/78.66/81.10	73.77/74.00/74.47	**78.25**
Endpoint reductions (Proposition 1)				
Endpoint (i): trust expected outputs†	84.15/85.98/86.59	74.39/75.61/79.88	72.37/73.54/73.77	76.97 -1.28
Endpoint (ii): discard expected outputs‡	75.61/75.61/76.22	66.46/67.68/68.29	72.60/72.60/73.07	71.56 -6.69
Component simplifications				
w/o adaptive κ (fixed tuned κ)	82.93/85.37/85.98	74.39/76.22/79.88	72.60/72.60/73.54	76.64 -1.61
w/o δ factor in κ	83.54/85.98/85.98	73.17/76.22/79.88	72.83/73.07/73.77	76.51 -1.74
w/o ρ factor in κ	82.93/85.37/85.98	75.00/77.44/79.88	73.07/73.07/74.00	77.00 -1.25
w/o per-view $L^{(i)}$ (single L)$^\maltese$	84.15/85.98/85.98	74.39/77.44/78.66	72.83/73.07/73.54	77.12 -1.13
single L via Jaccard on $B^{(n+1)}$	84.76/85.98/86.59	75.00/78.05/79.27	72.13/72.13/72.60	77.30 -0.95
single L via cosine on $B^{(n+1)}$	85.37/86.59/86.59	75.00/78.66/79.27	72.13/72.37/72.37	77.50 -0.75
w/o smoothing (λ=0, score fusion only)	84.76/85.98/86.59	74.39/76.83/79.27	73.07/73.30/73.77	77.41 -0.84
w/o sigmoid normalization	84.76/85.37/85.98	76.22/78.66/79.27	73.07/73.07/74.00	78.01 -0.24

† $\kappa\to\infty$, λ=0: reduces to ACES-O. ‡ κ=0 (tuned λ): reduces to cross-code agreement scoring.
$^\maltese$ $\{L^{(i)}\}$ replaced by one shared L from pairwise output match $S_{k,k'} = \frac{1}{m}\sum_j \mathbf{1}[Y_{k,j}{=}Y_{k',j}]$.

sampling on the codes axis preserves the correct-candidate ratio), with bands showing ±1 std over three seeds. At full data N$^+$PC leads ACES-O alone by 5.4% Pass@1; under 90% code-removal the gap widens to 11.7% (ACES-O alone collapses by 11 points to 63.1 while N$^+$PC loses only 5 to 74.8), and under 90% test-removal it stays at 8.2%. N$^+$PC's standard deviation stays at or below 1.4% across all conditions, while ACES-O alone's climbs to 2.4% at the heaviest code-removal. Two mechanisms produce this: $(n{+}1)$-view aggregation compensates for the missing (code $\times$ test) cells, and adaptive κ absorbs the seed variance the inner alone would propagate.

(ii) N$^+$PC's gains land in the regimes the motivation predicts. Fig. 7c stratifies tasks by tests accuracy (the fraction of LLM-generated test outputs agreeing with a correct candidate's outputs) and plots N$^+$PC's per-bin gain over ACES-O alone. The gain generally decreases as tests accuracy rises: N$^+$PC adds 21.6% in the bottom bin ($<20\%$), drops to zero across the 40% to 80% middle, and contributes a residual 2.3% at the top. This reproduces the motivation: the LLM expected output is informative only when the unstated code is itself plausible, and N$^+$PC's adaptive κ leans away from it precisely when that condition fails. The same regime-localization holds under both inners, so it reflects N$^+$PC's mechanism, not the inner scorer's form.

C.4 Pass@k Ablation Breakdown

§ 4.3 summarizes the endpoint and component ablations on dataset-averaged Pass@1 in Tab. 2. Tab. 3 extends this with the full Pass@1/2/5 breakdown across all three datasets and adds two further single-L code-similarity variants (Jaccard and cosine on $B^{(n+1)}$).

The four conclusions of § 4.3 carry over to Pass@2 and Pass@5 in dataset-averaged results. Both endpoint asymmetries persist throughout (with (i) below showing the between-endpoint one widens at higher k); inside adaptive κ, δ continues to outweigh ρ on average at every k ($1.74/1.33/0.84$ versus $1.25/1.12/0.77$ avg-Pass@$\{1,2,5\}$); per-view $L^{(i)}$ and the Laplacian smoothing each yield positive average gains throughout. Beyond these reaffirmations, the extended breakdown surfaces two observations the main-paper table does not directly show.

(i) The between-endpoint asymmetry widens at higher k. The trust-expected endpoint costs $1.28/1.37/0.64$ avg-Pass@$\{1,2,5\}$, while the discard-expected endpoint costs $6.69/7.79/8.19$ (a discard-to-trust ratio growing from $\sim5\times$ at k=1 to $\sim13\times$ at k=5). At higher k, the trust-expected endpoint's gap to the full method shrinks because a wider top-k window forgives some ranking

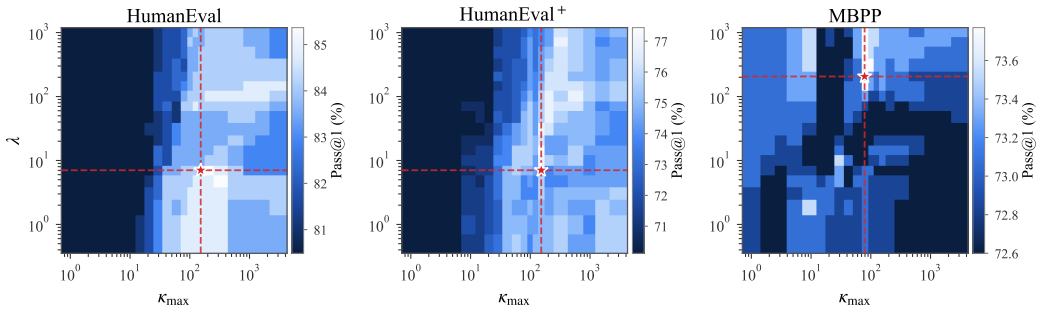


Figure 8: Pass@1 sensitivity to $(\lambda, \kappa_{\max})$ for N$^+$PC with the ACES-O inner. Red crosshair and star mark the reported $(\lambda, \kappa_{\max})$: $(\lambda{=}7, \kappa_{\max}{=}150)$ for HE/HE$^+$ and $(\lambda{=}200, \kappa_{\max}{=}80)$ for MBPP.

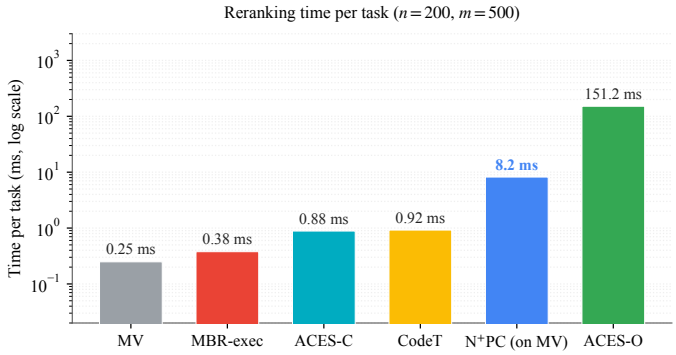


Figure 9: Per-task reranking time on MBPP ($n{\approx}200$, $m{\approx}500$), single GPU, log scale. N$^+$PC is shown on top of the MV inner (matching the configuration of Fig. 5).

imperfection; the discard-expected endpoint cannot benefit similarly, since it lacks the unstated-code reference altogether, so its gap actually widens with k.

(ii) The per-view formulation is what matters, not the pairwise similarity. Collapsing $\{L^{(i)}\}$ into a single L derived from Jaccard or cosine similarity on $B^{(n+1)}$ (in addition to the output-match variant in Tab. 2) loses 0.95 and 0.75 avg-Pass@1 respectively, alongside the 1.13 loss of the output-match variant. All three sit in a narrow band below the full method, so the loss traces to collapsing $(n{+}1)$ distinct graphs into one rather than to suboptimal similarity choice.

C.5 Hyperparameter Sensitivity

N$^+$PC uses two hyperparameters, λ and $\kappa_{\max}$. § 4.4 (iii) shows the HE$^+$ sensitivity heatmap; Fig. 8 extends it to all three datasets, sweeping Pass@1 over a 19×20 grid in $(\lambda, \kappa_{\max})$ with the ACES-O inner (per-panel color scales, since the three datasets differ by ~ 10 Pass@1 points).

The sensitivity landscape is wide and flat. HE and HE$^+$ each exhibit a broad high-Pass@1 plateau covering roughly an order of magnitude in λ and a half-order in $\kappa_{\max}$; MBPP shows a vertical band at $\kappa_{\max} \approx 80$ extending over $\lambda \in [150, 1000]$. Pass@1 is near-equivalent across a wide range of $(\lambda, \kappa_{\max})$ inside these regions, so the reported pair (red crosshair in Fig. 8) is an interior point of the high-Pass@1 plateau on each dataset.

Two configurations: one shared by HE/HE$^+$, one for MBPP. HE and HE$^+$ share reranker inputs and differ only in evaluation labels (§ 4.2 (iii)), so a single $(\lambda, \kappa_{\max})$ pair applies to both; MBPP receives one separate pair given its distinct candidate distribution. We use $(\lambda{=}7, \kappa_{\max}{=}150)$ on HE/HE$^+$ and $(\lambda{=}200, \kappa_{\max}{=}80)$ on MBPP.

C.6 PER-TASK RUNTIME

Fig. 9 reports per-task reranking time on MBPP ($n{\approx}200$, $m{\approx}500$). All methods run on the same single GPU; tasks are processed sequentially, with within-task work parallelized where the method admits it. Timing starts from the host-side output hashes (H for the n candidates and h_{exp} for the LLM expected output); the pass matrix $B = \mathbf{1}[H = h_{\text{exp}}]$ is built inside the timed region for any method that needs it, and upstream costs (generation, execution, hashing) are excluded as they are shared across rerankers. Numbers are the median across the MBPP task set.

The methods cluster into three tiers. The four lightweight baselines (MV, MBR-exec, ACES-C, CodeT) each reduce to a short sequence of vectorized GPU operations and finish in well under one millisecond per task. N$^+$PC sits in the middle tier at ≈ 8 ms, where the spectral aggregation (building the $(n{+}1)$ per-view Laplacians and solving the regularized linear system) dominates over the lightweight MV inner. ACES-O, with its 90 Adam iterations, sits in the upper tier at ≈ 150 ms. Every method completes a task in well under one second, and reranking remains negligible relative to the upstream execution step that produces the input hashes.