

Duel-Evolve:

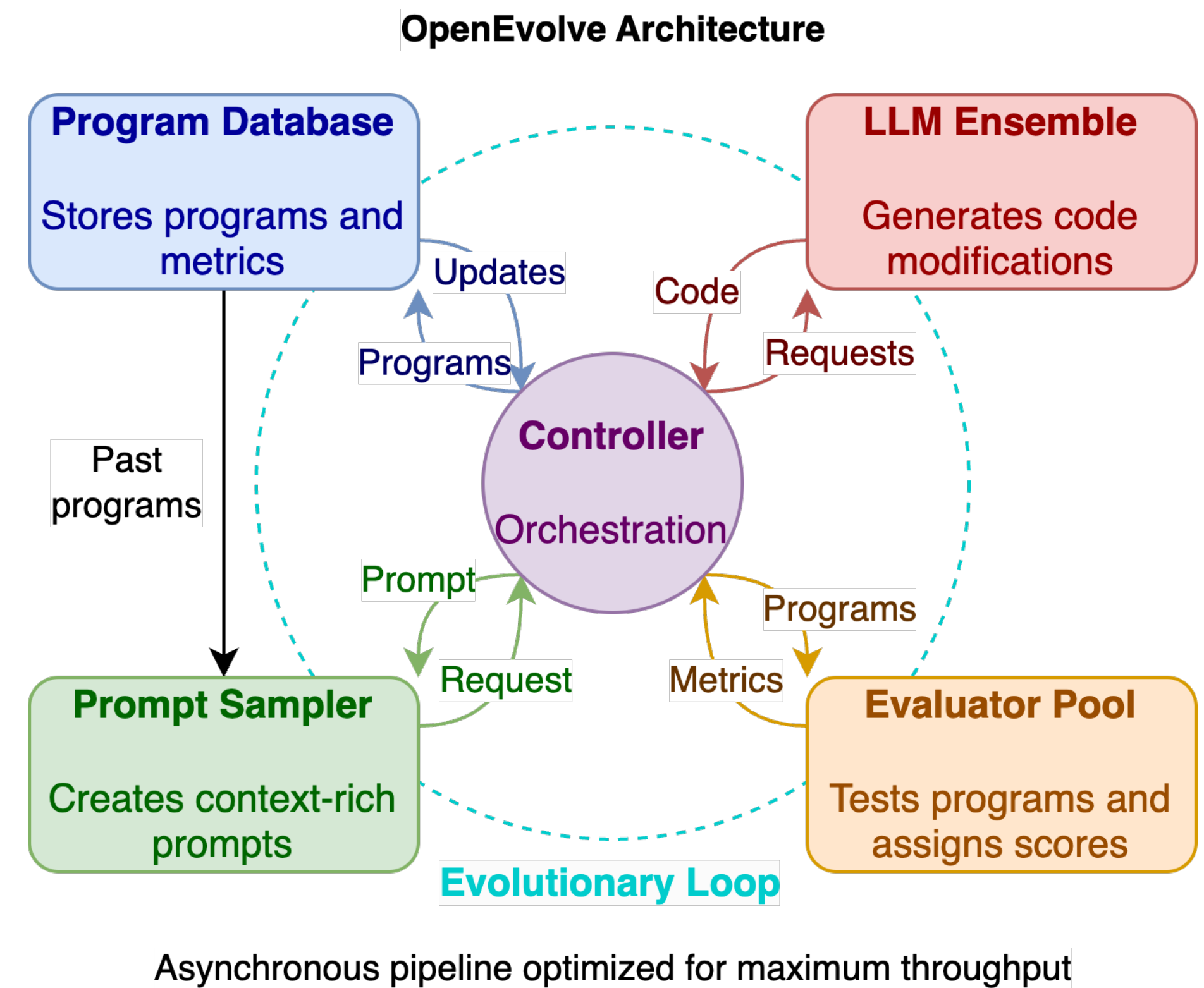
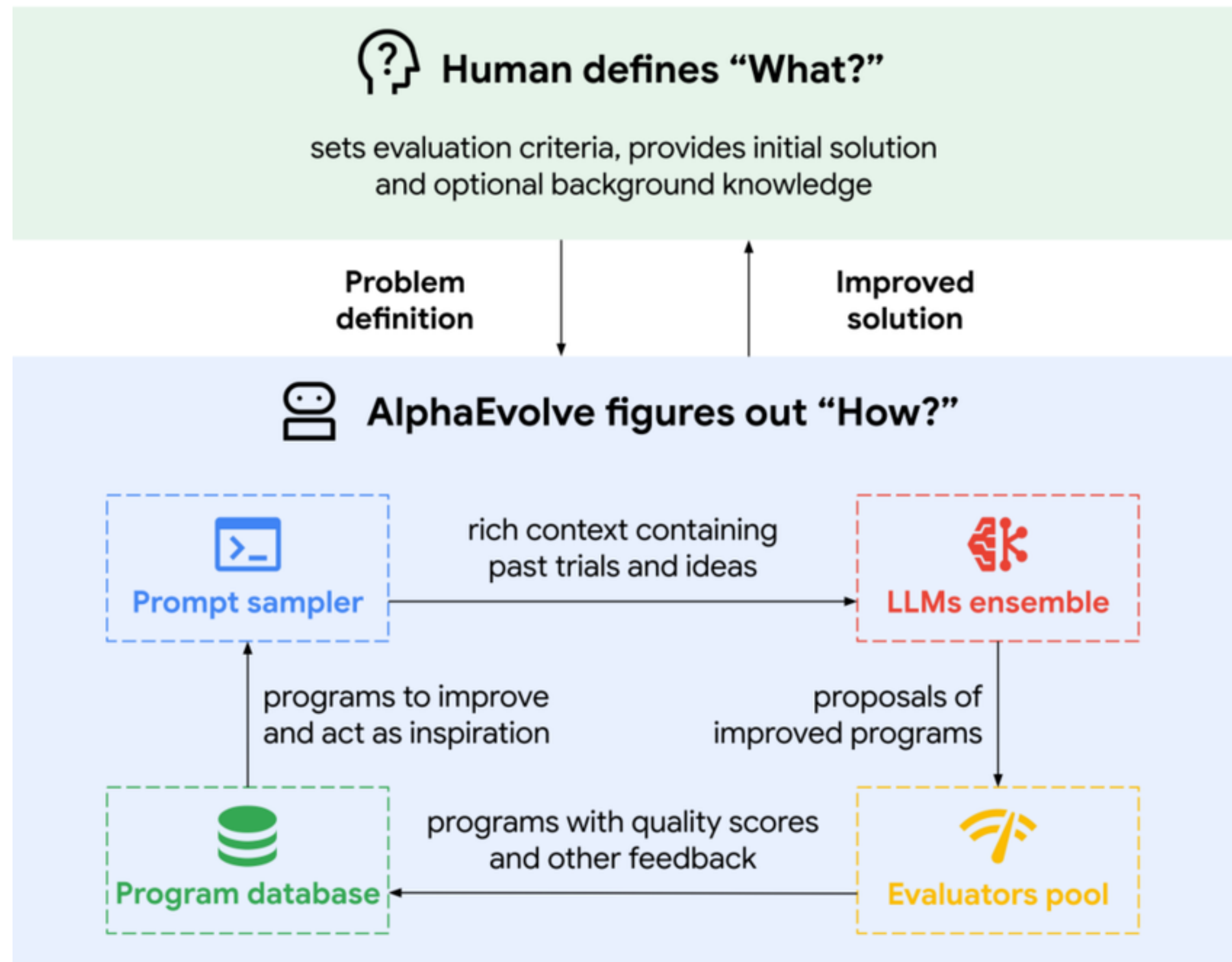
Reward-Free Test-Time Scaling via LLM Self-Preferences

Sweta Karlekar*, Carolina Zheng*, Magnus Saebo*, Nicolas Beltran-Velez, Shuyang Yu, John Bowlan, Michal Kucer, David Blei

Agenda

- AlphaEvolve
- When does AlphaEvolve not work
- Duel-Evolve
- Experiments + Results

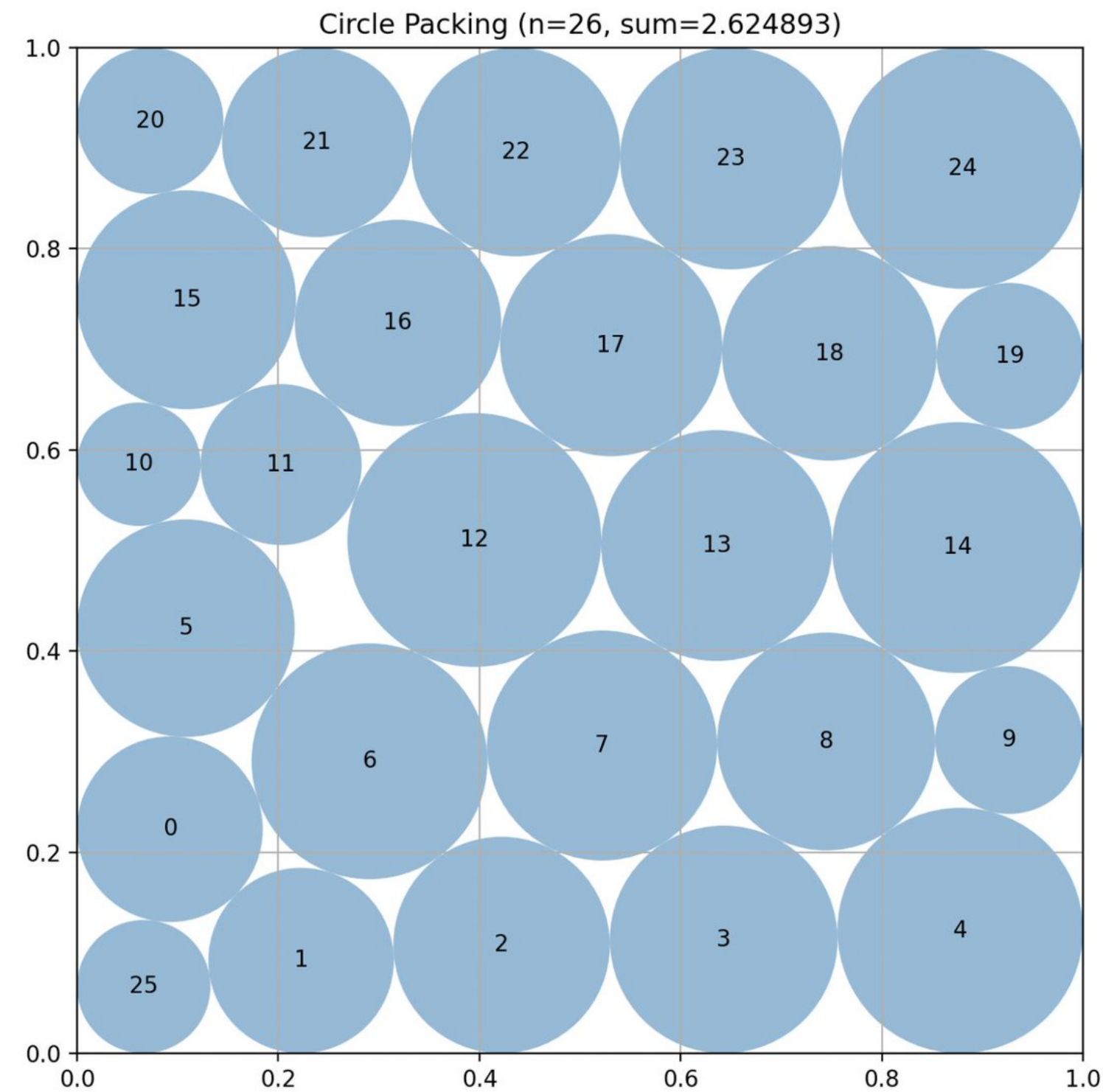
What is AlphaEvolve?



AlphaEvolve Example

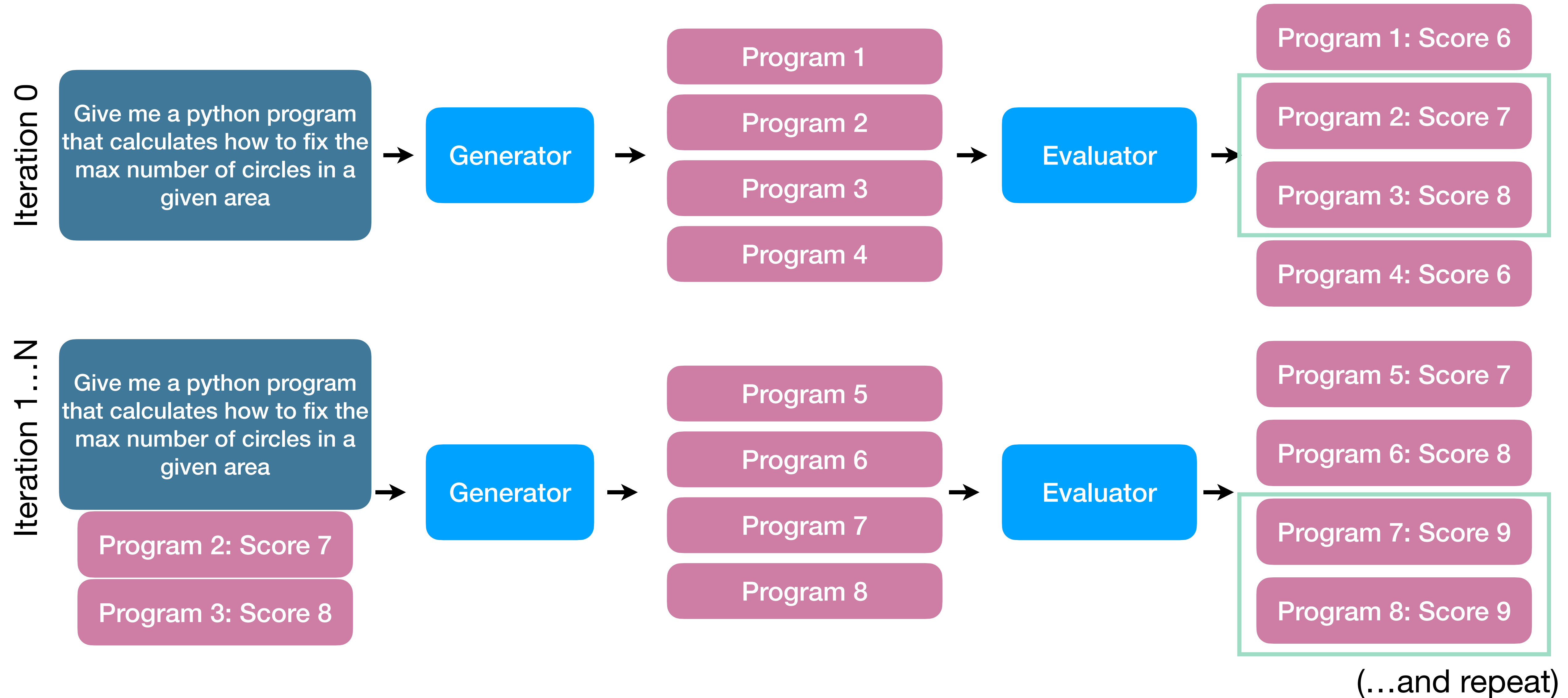
Circle Packing Problem:

Give me a python program that calculates how to fix the max number of circles in a given area



- *Space of optimization:* python programs
- *Score to optimize:* number of circles

AlphaEvolve (simplified)



Benefits of AlphaEvolve

- 1) AlphaEvolve takes advantage of **latent abilities/knowledge** of the LLM. It may not know how the optimal answer zero-shot, but it can incrementally improve upon existing solutions.
- 2) Unlike CoT, we can **continue evolutionary cycles** if we haven't gotten the right answer. CoT breaks down a single task into chunks, but once it's gotten to the end, it's done. AlphaEvolve can continue to evolve and improve its answer (until it plateaus).

Issues with AlphaEvolve

- 1) AlphaEvolve needs a numeric score to optimize. This isn't feasible in a lot of cases:
 - *Multiple heuristics:* code efficiency can be measured by runtime, theoretical efficiency, hardware efficiency, energy efficiency, scalability.
 - *Sparse reward:* binary settings like theorem solving is either 0 (unsolved) or 1 (solved), which doesn't give AlphaEvolve signal on incremental improvements
 - *Subjectivity:* summarization, creative writing, translation, etc. cannot easily be converted into numerical scores to optimize on.
- 2) Further, it takes time and effort to create an evaluation scheme or rubric, even if it is possible for a given task.

Duel-Evolve

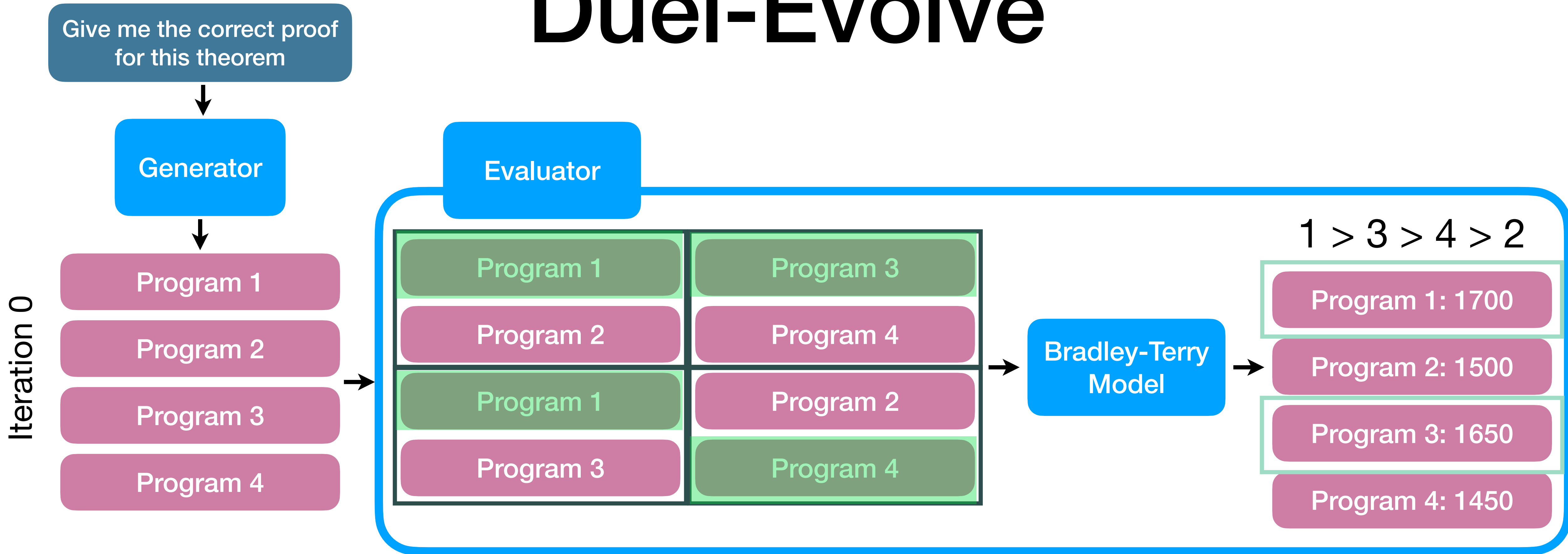
We can answer both of the previous issues is using **LLM pairwise self-preferences!**

For many tasks, it is unclear how to construct an agreed-upon universal real-valued score. However, in most cases, it is much easier to tell which of two solutions is better.

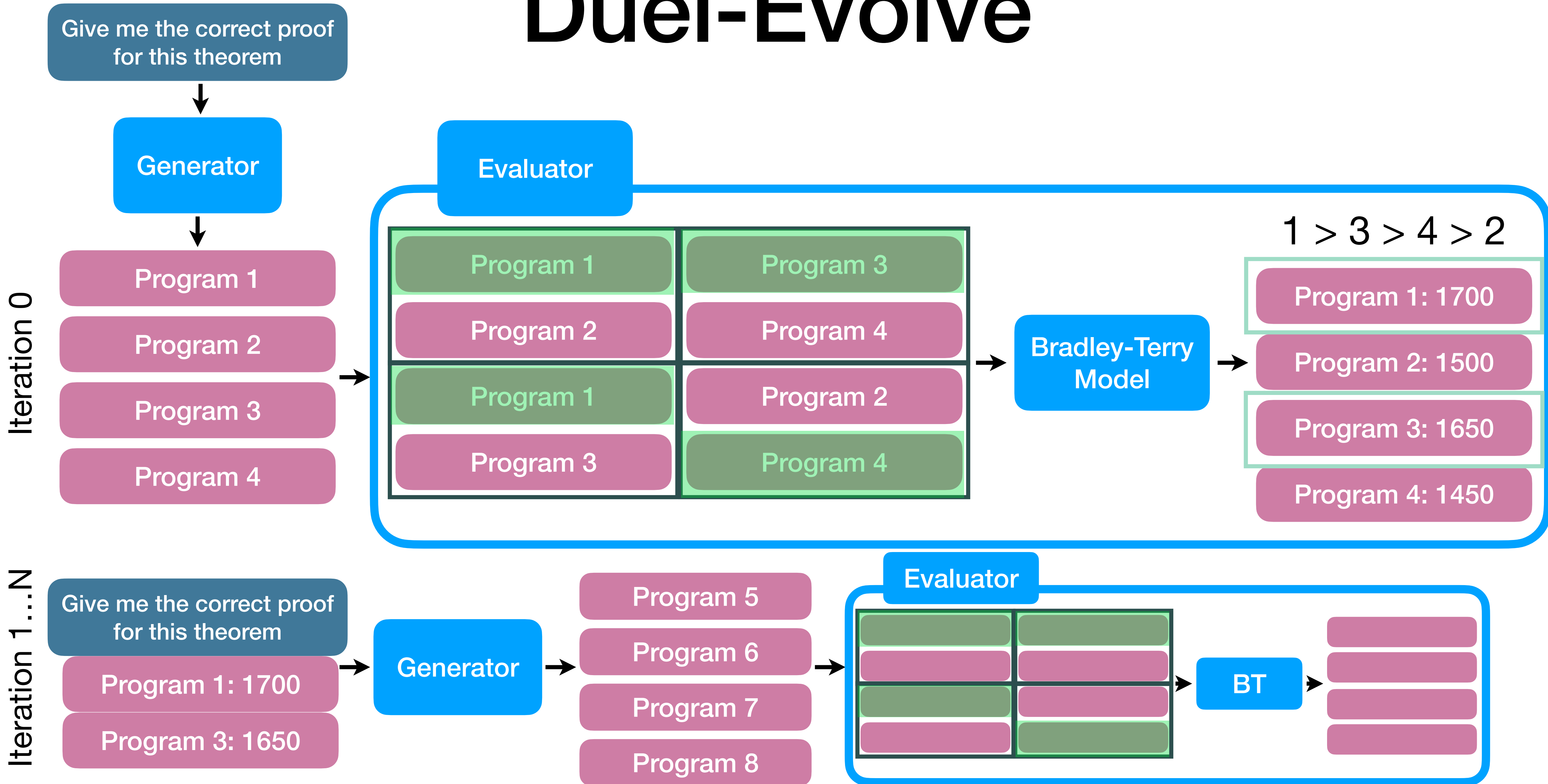
This again takes advantage of the latent abilities of the LLM; it may not immediately produce the right answer, but it is more likely to be able to tell which of two solutions is *more correct*.

We can then aggregate these pairwise rankings using a ranking model like Bradley-Terry to get a global score that we can optimize over.

Duel-Evolve



Duel-Evolve



Bradley-Terry Model

Let there be items $i \in 1, \dots, n$.

Each item has a latent parameter $\theta_i \in \mathbb{R}$.

For a comparison between items i and j , define:

$$y_{ij} = \begin{cases} 1 & \text{if } i \text{ is preferred over } j \\ 0 & \text{if } j \text{ is preferred over } i \end{cases}$$

The Bradley-Terry model assumes:

$$P(i \succ j) = \frac{\exp(\theta_i)}{\exp(\theta_i) + \exp(\theta_j)}$$

Fitting a Bradley-Terry Model

We are given a dataset of pairwise comparisons $\mathcal{D} = \{(i_k, j_k, y_k)\}_{k=1}^N$.

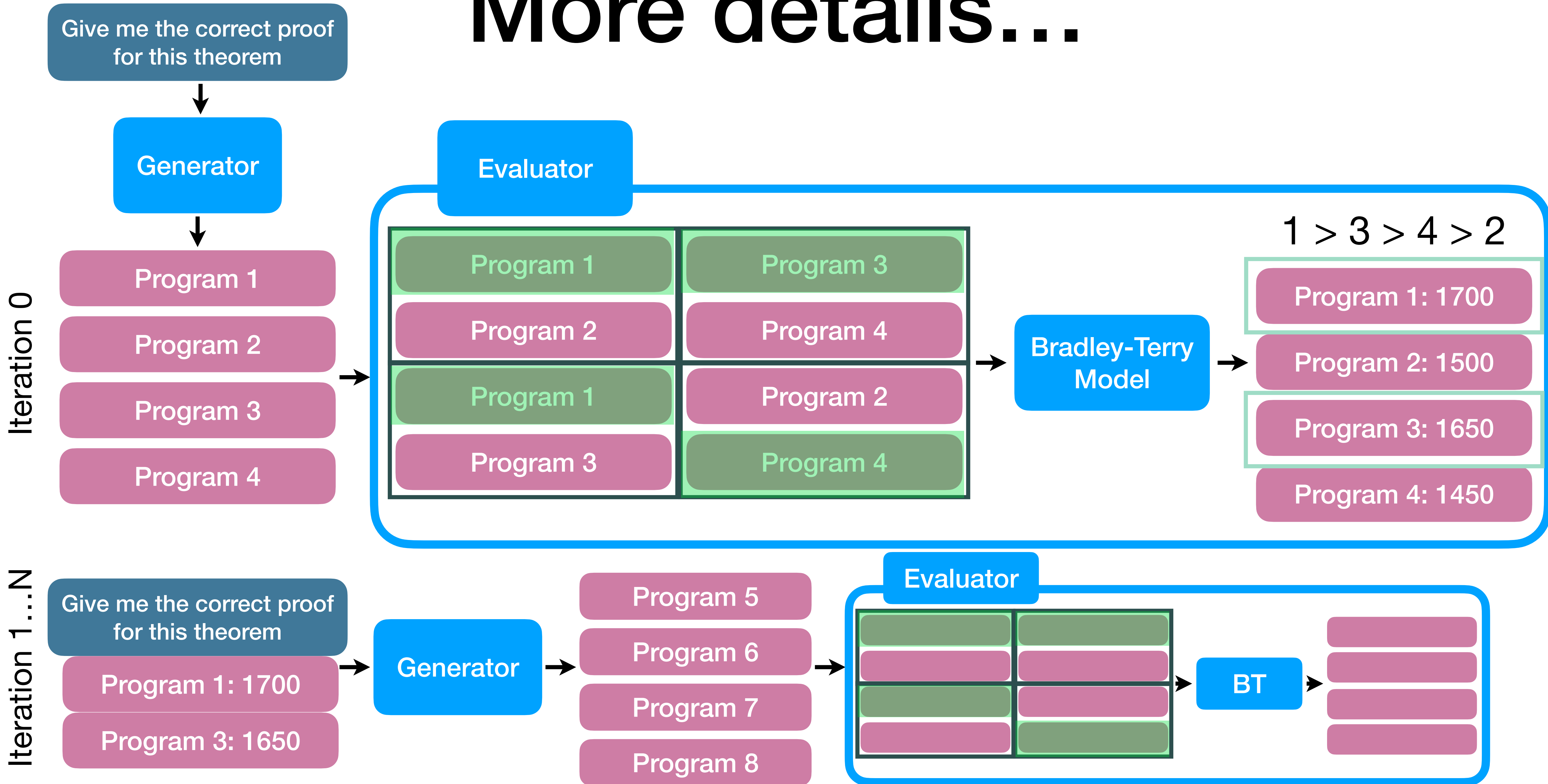
Assume independent over comparisons. The log-likelihood becomes:

$$\ell(\theta) = \sum_{k=1}^N \left[y_k \log \sigma(\theta_{i_k} - \theta_{j_k}) + (1 - y_k) \log(1 - \sigma(\theta_{i_k} - \theta_{j_k})) \right].$$

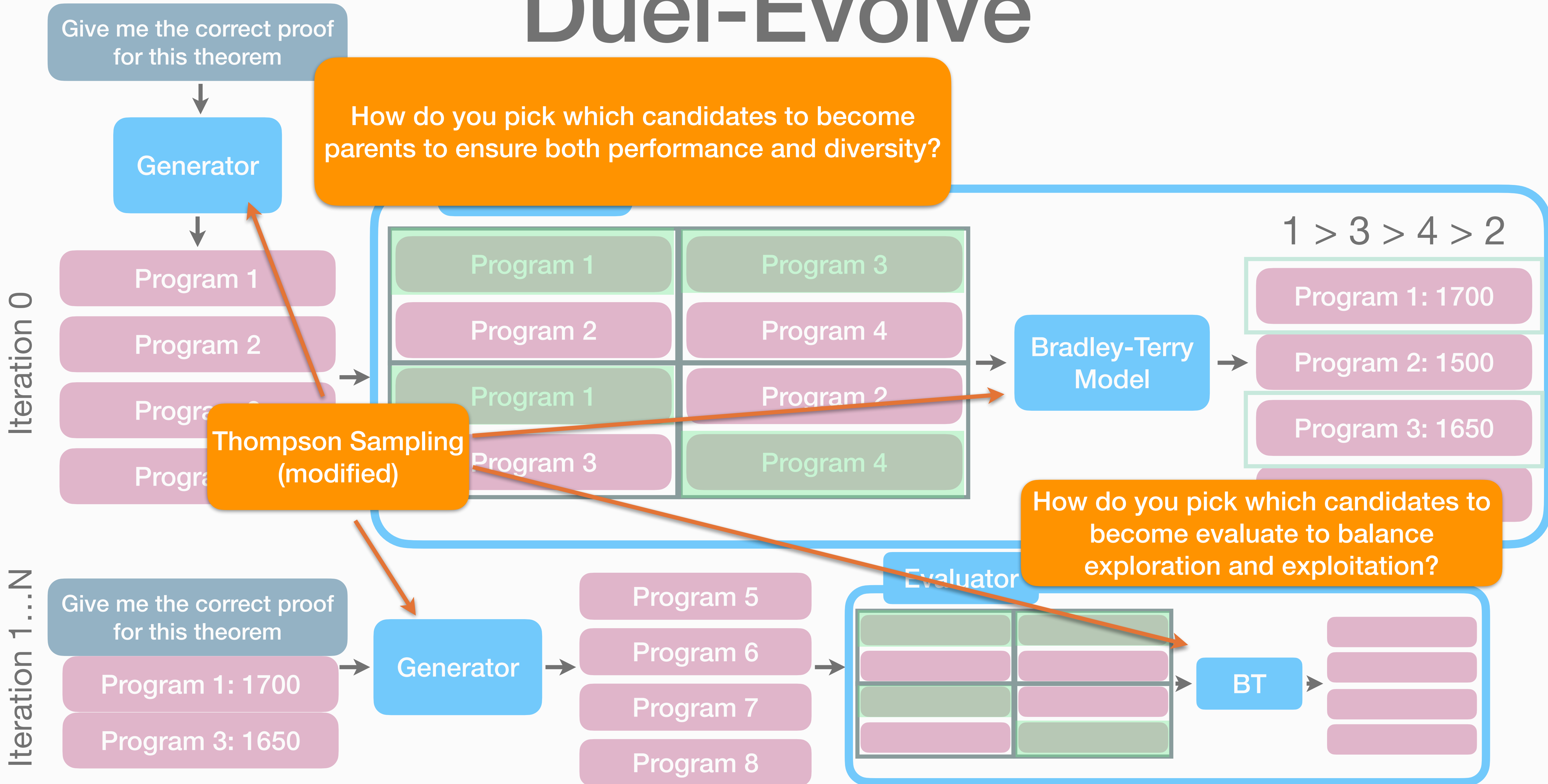
The model is invariant to adding a constant, so we typically impose a constraint such as fixing one item (e.g. $\theta_1 = 0$).

Each θ_i is a latent score, and only differences between the latent scores matter. The model assumes transitivity.

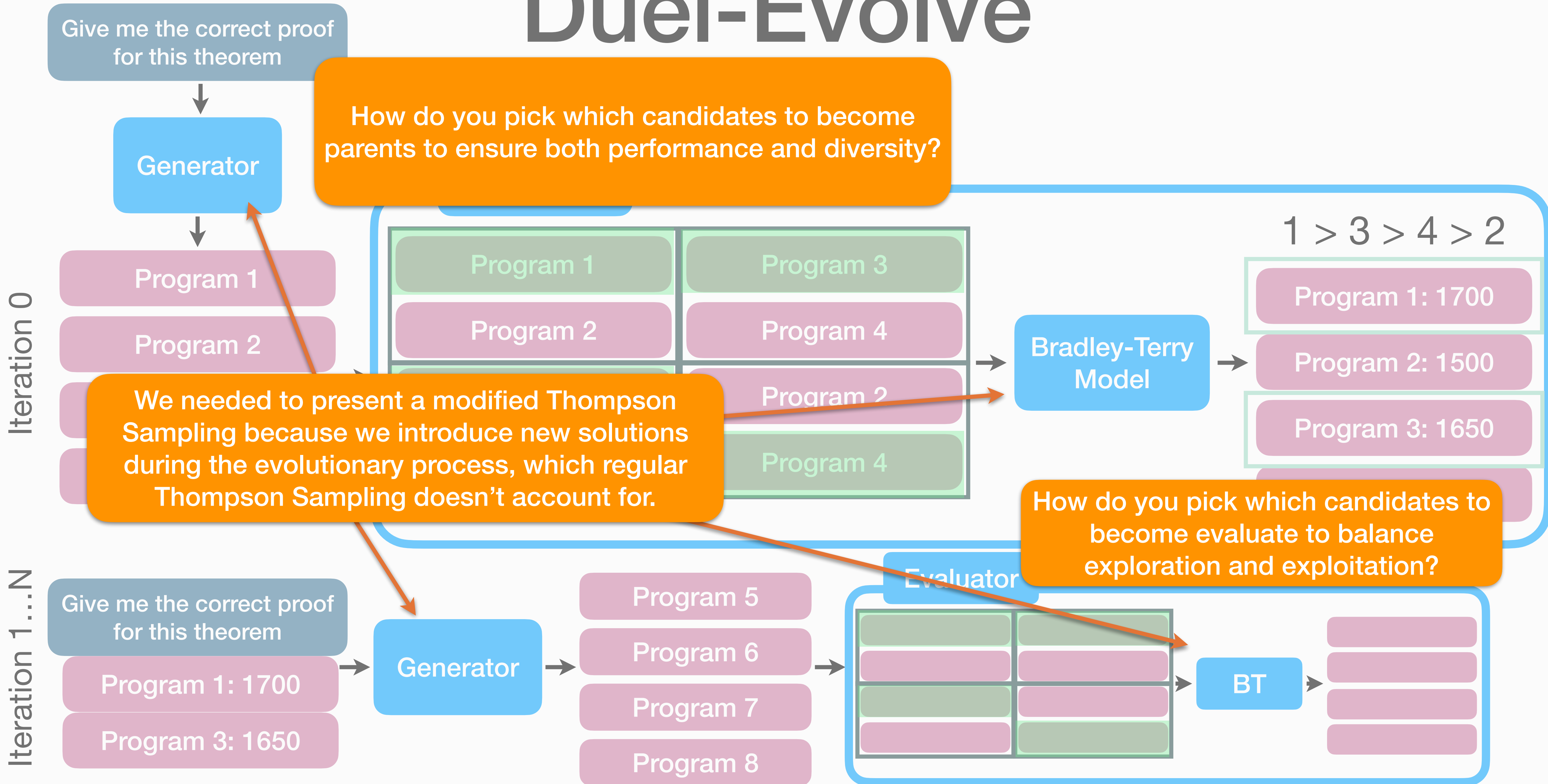
More details...



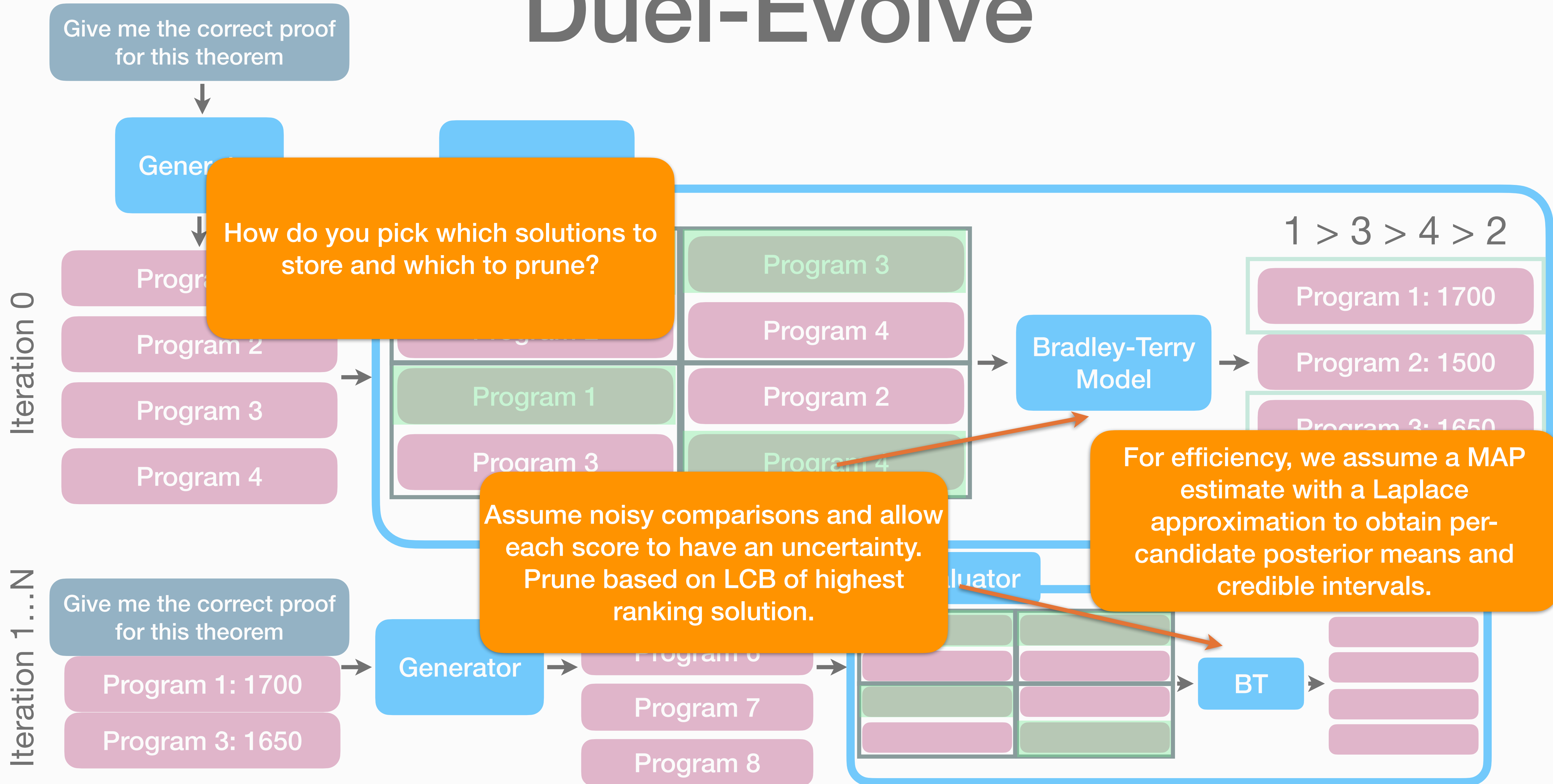
Duel-Evolve



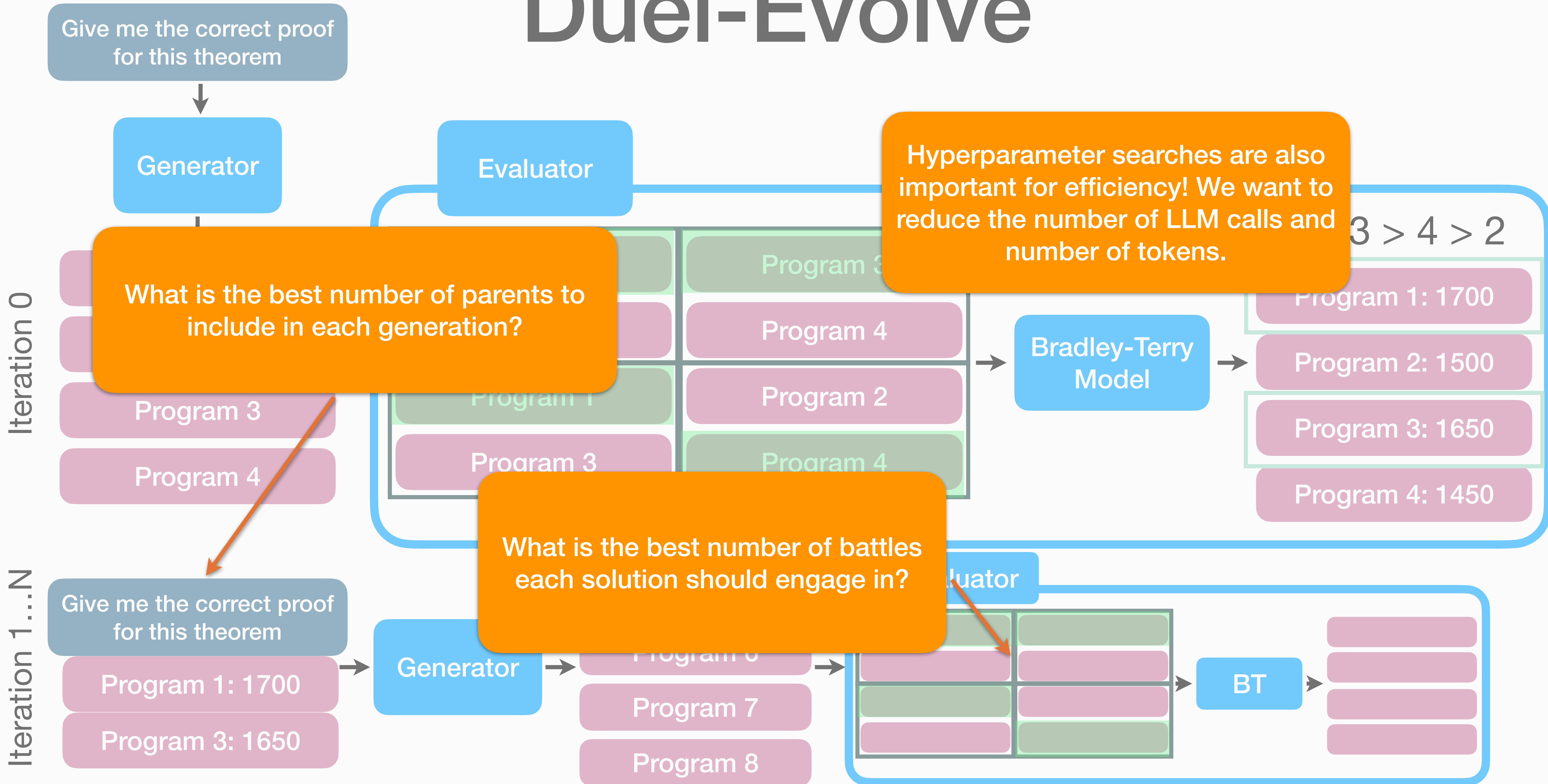
Duel-Evolve



Duel-Evolve



Duel-Evolve



Experiments

- Task 1: MathBench

Supposed 15% of x equals 20% of y . What percentage of x is y ?

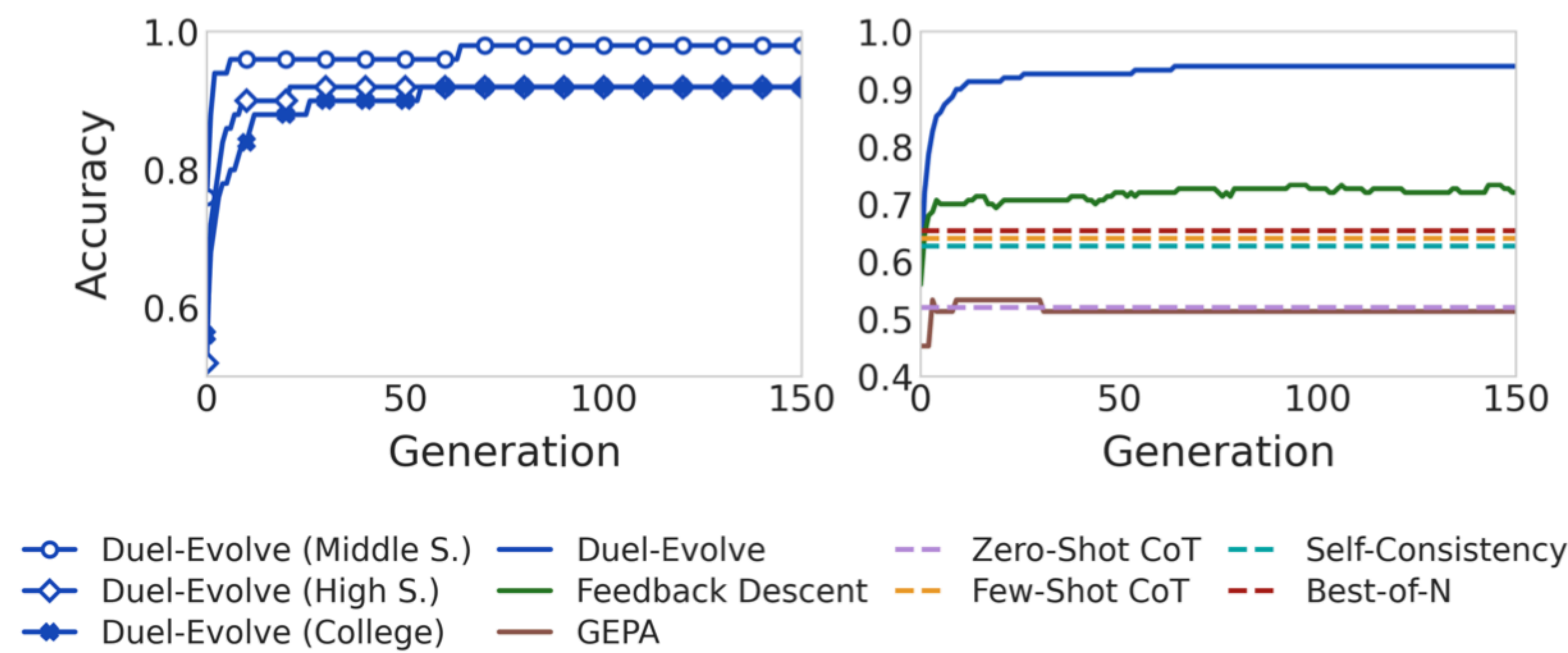
A. 75% B. 5% C. 35% D. 300%

- Task 2: LiveCodeBench

You are given a positive integer array `nums`. Return the total frequencies of elements in `nums` such that those elements all have the maximum frequency.

Duel-Evolve Results

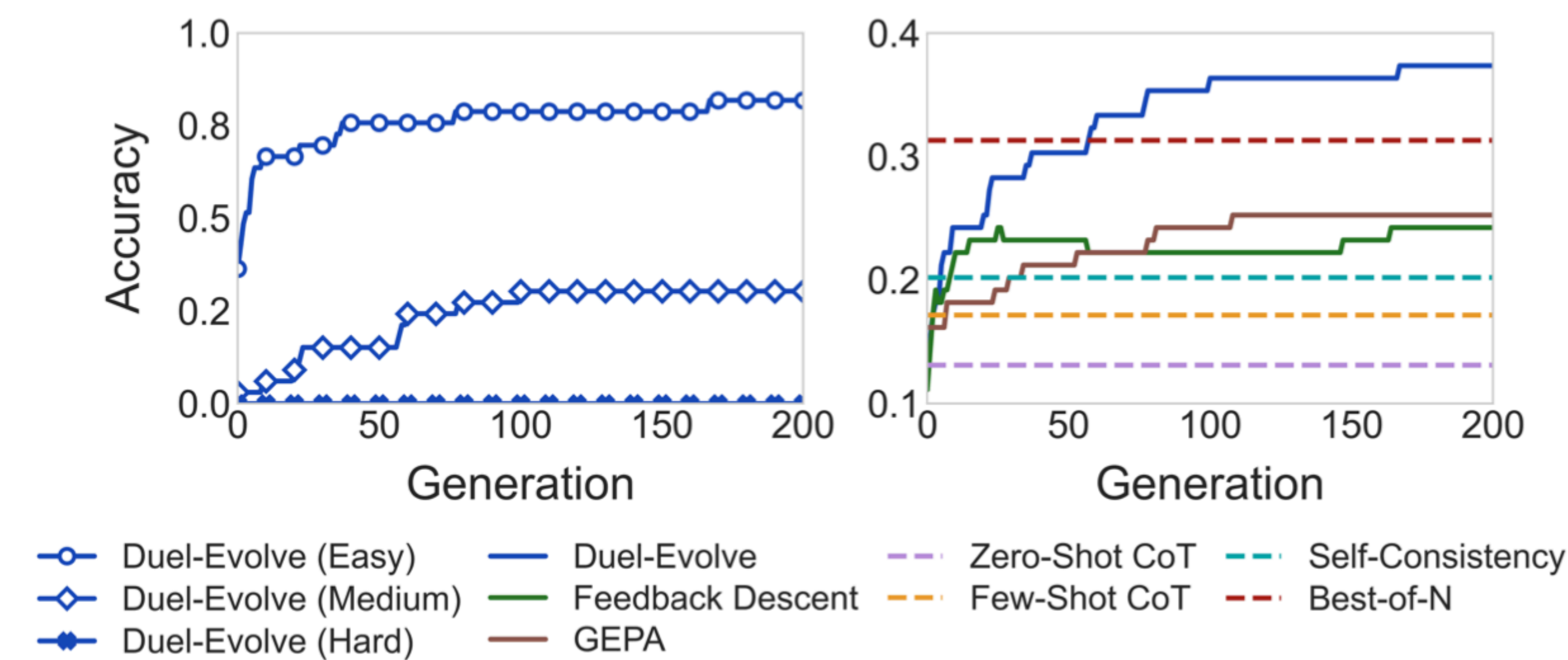
Gemma-2 4b model.



Method	Accuracy (%)
Zero-shot CoT	57.3
Few-shot CoT ($k=8$)	64.0
Self-consistency	62.7
Best-of- N	65.3
GEPA	51.3
Feedback descent	72.0
DUEL-EVOLVE	94.0

Duel-Evolve Results

Gemma-2 27b model (int4 quantized).



Method	Accuracy (%)
Zero-shot CoT	13.1
Few-shot CoT ($k=3$)	16.2
Self-consistency	20.2
Best-of- N	31.3
GEPA	25.3
Feedback Descent	24.2
DUEL-EVOLVE	37.4

Current Status



<https://arxiv.org/pdf/2602.21585>

- Accepted at the Recursive Self-Improvement (RSI) Workshop at ICLR 2026
- Currently working on next iteration for ICLR 2027 submission:
 - Include more math and code tasks, plus a creative writing benchmark (Arena-Hard)
 - Generic prompt Chatbot-Arena style evaluation against base models
 - (Any other suggestions?)