SELECTING WHAT TO TRANSMIT: CROSS-GENERATIONAL IN-CONTEXT REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

In-context reinforcement learning promises that a trained network can continue to improve on a new instance by reading its own attempts, without updating its parameters. Prior work seeks better inference performance while maintaining efficiency, typically by accumulating more information in the context window. This is a trade-off: more information in the context window raises the computational cost of inference. Inspired by the intergenerational accumulation of knowledge in human societies, we move from within-generation to cross-generational inference. To this end, we propose CGIRL, which partitions inference into successive generations. Within each generation, the policy reads a fixed short window. Across generations, MAP-Elites, an evolutionary algorithm, selects the transmitted experience from candidates archived by their return dynamics, rather than transmitting the full history or a single incumbent. On the travelling salesperson problem and a sequence-recall task, experiments show that CGIRL achieves state-of-the-art performance and efficiency.

1 INTRODUCTION

A policy retrained for every new instance does not scale. In-context reinforcement learning offers an alternative: train once, and let further improvement occur only through the context that the policy reads (Laskin et al., 2023; Lee et al., 2023; Zisman et al., 2024; Son et al., 2025). Language models made this setting familiar (Brown et al., 2020), and transformers have since been shown to implement in-context RL from learning histories, including under supervised pretraining, model-based distillation, and large-scale meta-training (Chen et al., 2021; Lee et al., 2023; Son et al., 2025; Nikulin et al., 2024; Wang et al., 2025). If a single network is to continue improving on a new instance without updating parameters, the same mechanism must remain effective after many attempts, not only after a few demonstrations. How context carries test-time improvement is therefore a central question.

Prior work seeks to improve inference performance while maintaining efficiency, typically by accumulating additional information in the context window. Autoregressive in-context RL grows a history of trials and attends to all of them (Laskin et al., 2023; Zisman et al., 2024). Retrieval methods fetch similar sub-trajectories so that the window need not store an entire episode (Schmied et al., 2025). These methods differ in how the window is filled, yet they share a limitation: increasing the information in the window raises the computational cost of inference, as history cross-attention grows with the number of trials and occupies the channel used to read the instance. In practice, constructing a richer context often amounts to a longer memory.

Inspired by the intergenerational accumulation of knowledge in human societies (Cook et al., 2024), we move from within-generation to cross-generational inference. We recast a single continuous rollout as staged inference, where each stage is a generation. Within a generation, the process is standard in-context RL and the policy reads a fixed short window. When the stage ends, the window is reset, and only one experience is transmitted. When inference crosses generations, the number of attempts can grow without letting the window grow with the total number of steps. Within-generation inference is ordinary in-context reasoning. Cross-generational inference is the transfer between stages. Figure 1 contrasts the two protocols.

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.

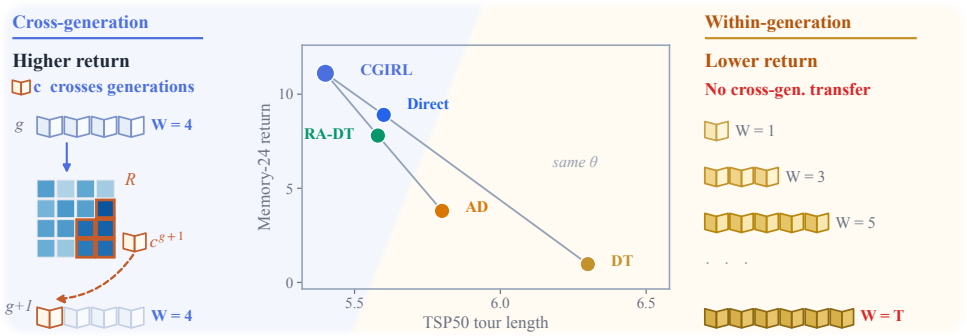

Figure 1: Cross-generation versus within-generation under the same θ. Left: generations exchange only the transmitted experience c, selected from the region $\mathcal{R}$, while the window stays fixed. Right: one generation grows the window with the number of trials and has no cross-generation transfer. Center: five operators on TSP50 and Memory-24.

We propose CGIRL to implement this process at test time. Attempts on the same instance are partitioned into generations that share one experience $c^{(g)}$ and a short window, and c is replaced only between generations. If only the current best is passed into the next stage, the population tends to collapse onto a single behaviour. CGIRL therefore uses MAP-Elites in the transfer step. It archives candidates by their within-generation return dynamics and selects the transmitted experience $c^{(g+1)}$, rather than the full history or a single incumbent. The archive selects what to transmit. It is not a separate search objective. Experimental results demonstrate that CGIRL outperforms a single-generation window, concatenated history, and retrieval.

Our contributions are as follows.

- We extend within-generation in-context RL to cross-generational inference that keeps the window fixed and replaces only the transmitted experience c, improving inference while controlling its computational cost.
- We introduce MAP-Elites into the transfer step and choose the transmitted experience from within-generation return dynamics rather than from a single incumbent. Later generations then condition on experience that continued to improve.
- We compare CGIRL with representative within-generation in-context RL methods on TSP and Memory. The same policy outperforms a single-generation window, concatenated history, and retrieval.

2 Related Work

In-context and meta-reinforcement learning. A line of work trains a policy to improve from its own interaction history without further gradient updates. RL^2 and related meta-RL methods encode the learning algorithm in recurrent weights, and later agents scale the same idea to long-context transformers (Duan et al., 2016; Wang et al., 2016; Grigsby et al., 2024a;b). Algorithm Distillation casts in-context RL as sequence modeling over learning histories (Laskin et al., 2023), and later work shows that such in-context improvement can emerge from noise distillation or from distilling model-based planners (Zisman et al., 2024; Son et al., 2025). Decision Transformer and supervised in-context RL methods read trajectories as context and act without updating parameters (Chen et al., 2021; Lee et al., 2023). We keep this setting. The remaining question is what should be transmitted once the context window can no longer grow.

Context construction. Autoregressive in-context RL retains every trial in one window and attends to the full history (Laskin et al., 2023; Chen et al., 2021). Retrieval-augmented methods store fragments in an external memory and fetch similar sub-trajectories, so the window need not hold an entire episode (Schmied et al., 2025). Decision Transformer and related sequence models likewise grow the context with completed trajectories (Chen et al., 2021; Dong et al., 2025; Huang et al.,

2024; Lee et al., 2023). These families improve inference by putting more information into context. History cross-attention then grows with the number of trials and occupies the channel used to read the instance. We do not lengthen the window. Within a generation the policy reads a fixed short window; between generations only one compact experience is replaced.

Cross-generational transfer. Cultural accumulation studies how agents combine individual exploration with information passed across generations (Cook et al., 2024). That line of work still copies the previous generation as a demonstration, and anneals demonstration visibility so later trials can act independently. We keep the within-generation visibility schedule. Generational training in RL likewise typically copies a single incumbent (Duan et al., 2016; Laskin et al., 2023). Copying only the current best tends to collapse later generations onto one behaviour. MAP-Elites was proposed to keep diverse elites on a descriptor grid rather than one incumbent (Mouret & Clune, 2015). Recent quality-diversity methods still fill that grid with reinforcement learning (Faldor et al., 2024; Faldor & Cully, 2025). We use it only as a test-time transfer operator: what crosses generations is chosen from within-generation return dynamics, not by copying the previous generation or a single incumbent.

3 PRELIMINARIES

This section introduces the notation of in-context reinforcement learning and the MAP-Elites procedure used later. The cross-generational partition is defined in Section 4.

3.1 IN-CONTEXT REINFORCEMENT LEARNING

In-context reinforcement learning improves on a new instance by reading context, without updating parameters. On a fixed instance x, a conditional policy $\pi_\theta(\cdot \mid x, c)$ produces a trajectory given the current experience c, where θ denotes the policy parameters. One attempt is a trial and corresponds to one piece of experience: starting from x, the policy selects actions step by step and obtains a trajectory τ and a score s. Over consecutive trials, completed experience may be read by later ones. The policy reads at most the most recent W pieces of experience at a time. This is the context window. Improvement comes only from replacing c. Training provides demonstrations so that the policy learns to read this form of input and can still complete x after the demonstration disappears. How context is read depends on the task. TSP uses a Transformer. Memory uses a GRU. Both instantiations are given in the appendix.

3.2 MAP-ELITES

MAP-Elites is an evolutionary algorithm that keeps diverse solutions on a descriptor grid instead of a single incumbent (Mouret & Clune, 2015). It specifies a fitness f and a descriptor $\phi : \mathcal{X} \to \mathcal{D}$, where $\mathcal{X}$ is the space of candidate solutions and $\mathcal{D}$ is the descriptor space, then stores at most one solution in each cell of a discrete grid

$$\mathcal{A} : \mathcal{D} \to \mathcal{X} \cup \{\emptyset\}, \tag{1}$$

where $\mathcal{A}$ denotes the archive and $\emptyset$ an empty cell. The loop samples an elite, mutates it, and assigns a cell by ϕ. An empty cell is filled, and an occupied cell is replaced when the new fitness is higher. Search therefore covers distinct regions of the feature space rather than shrinking along a single optimum.

4 METHOD

The method has three parts. Section 4.1 describes cross-generational inference at test time: the current generation reads $c^{(g)}$ and produces trajectories. Section 4.2 uses MAP-Elites to archive candidates by descriptor and select elites from the region $\mathcal{R}$ as $c^{(g+1)}$. Section 4.3 separates training from inference. Training updates only the policy parameters, and inference updates only c. Figure 2 illustrates the procedure. The encoding of experience c is task-dependent. The rules for reading, archiving, and transfer are not.

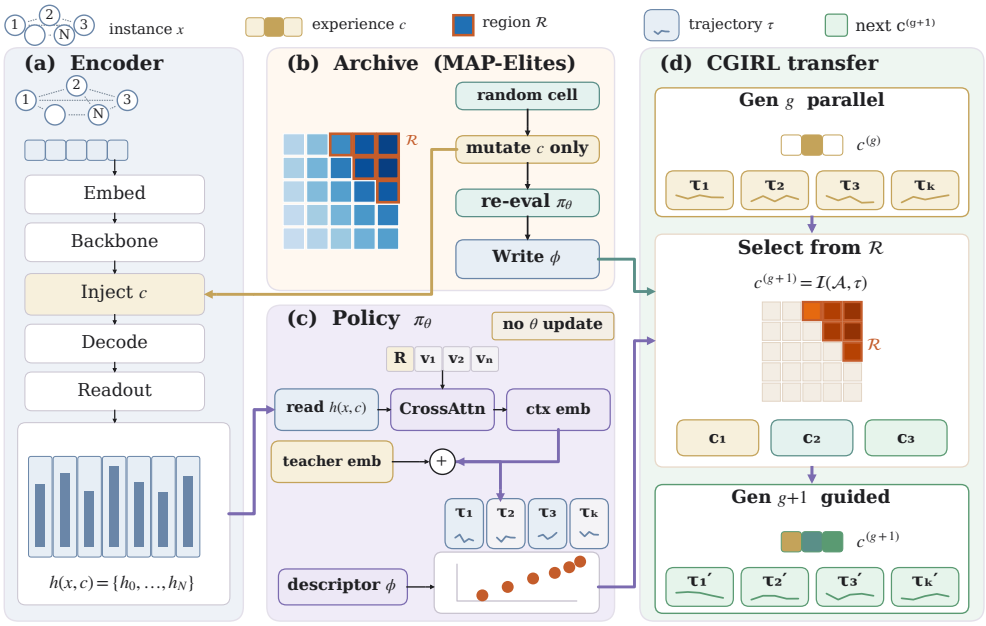

Figure 2: The architecture of CGIRL. (a) The encoder maps the instance x and the transmitted experience c to $h(x, c)$. (b) MAP-Elites archives by descriptor, and search mutates c only. (c) The policy reads $h(x, c)$ and the demonstration. (d) Elites from the region $\mathcal{R}$ become $c^{(g+1)}$. Visibility decreases over trials within a generation.

4.1 Cross-generational in-context reinforcement learning

The consecutive trials of Section 3.1 are partitioned into generations so that the window need not grow with the total number of steps. A generation consists of parallel agents, each running K trials. During that generation, the experience $c^{(g)}$ is held fixed: the current generation can read it but cannot change θ. The generation produces trajectories

$$\tau^{(g)} \sim \pi_\theta(\cdot \mid x,\, c^{(g)}), \tag{2}$$

and a vector of trial scores $s^{(g)} = (s_1^{(g)}, \ldots, s_K^{(g)})$. After it ends, the next context is

$$c^{(g+1)} = \mathcal{I}\big(\mathcal{A}^{(g)}, \tau^{(g)}\big), \tag{3}$$

where $\mathcal{I}$ denotes the transfer operator and $\mathcal{A}^{(g)}$ the archive at the end of generation g. A direct choice of $c^{(g+1)}$ is to pass the best individual of the current generation. Standard in-context learning places demonstrations in a single decoding pass. Here, a generation takes the place of that pass: members of generation g share one $c^{(g)}$ and run K consecutive trials on the same instance x.

Conditional reading. Both x and c enter π_θ only as inputs. The network encodes x and the injected c into a conditional representation $h(x, c)$ and then selects actions to produce $\tau_{1:K}$. Within a generation, $c^{(g)}$ is held fixed, and the embeddings are not rewritten at each decoding step. A new generation changes only the c in the input, not θ. Experience c may be a previous-generation tour, a hidden state, or another task-specific form. Where it enters the network, and how it is implemented, are task-dependent. In the default setting, we concatenate the observation embedding of c into the input that the encoder already uses to read demonstrations. Whether this embedding is concatenated into the observation or added to a cross-attention output, only the input channel changes.

Training uses demonstrations with controllable visibility so that the policy learns to read this form of input. At test time, those demonstrations are removed, and the same channel is filled by the previous generation's experience. Generation $g = 0$ has no previous generation and starts from a task-specific seed. Generation $g > 0$ reads $c^{(g)}$. Parallel agents share this one context and produce their own trajectories, which become candidates for $c^{(g+1)}$.

Within-generation development and visibility. Within a generation, the process is developmental: with $c^{(g)}$ fixed, the agents run K trials and obtain a score trajectory $s^{(g)} = (s_1^{(g)}, \ldots, s_K^{(g)})$. The previous generation is in a transmission phase: its experience is read in each trial of the current generation. If the demonstration remained visible, the state at the end of the generation could still depend on prompting, and the transmitted behaviour would not be independent. Within-generation visibility therefore follows

$$P(\text{vis}) = \frac{K - k - 1}{K - 1}, \tag{4}$$

where $P(\text{vis})$ denotes the probability that the demonstration is visible on trial k, and k is the trial index. The probability decreases with the trial index k; the demonstration is always visible on trial 0 and never visible on the last trial. Intermediate trials rely on it less and less, so the policy must move from following the demonstration to completing x on its own. End-of-generation evaluation uses the last unprompted trial. That trajectory and the full $s^{(g)}$ are written into the archive as candidates.

4.2 MAP-ELITES TRANSFER

If only the best individual of the current generation is passed on, the population collapses onto a single behaviour. The next generation then inherits a terminal optimum rather than how the generation improved. At test time, MAP-Elites archives by descriptor, search mutates c only, and elites are taken from the region $\mathcal{R}$ (weak early, strong late).

Archive. To keep diverse experience across generations on the same instance, equation equation 1 is specialised to a persistent archive on each test instance,

$$\mathcal{A} : \mathcal{D} \to (c, f), \tag{5}$$

where each cell stores one experience c and its fitness f. A new candidate that falls into an occupied cell replaces the occupant when its fitness is higher. The archive is kept across generations on the same instance. The first generation starts with an empty archive and a task-specific seed. After current-generation candidates are written, search is applied, and experience for the next generation is selected from the region $\mathcal{R}$.

Descriptor. The descriptor splits the score trajectory s at its midpoint and quantises the two halves onto a $B \times B$ grid using the task-inherent score upper bound $s_{\max}$:

$$\phi = \big(\text{bin}(s_{\text{early}}),\, \text{bin}(s_{\text{late}})\big) \in \{0, \ldots, B-1\}^2, \tag{6}$$

where s_{early} and s_{late} denote the mean score on the first and second halves of s, $\text{bin}(\cdot)$ maps a score into $\{0, \ldots, B-1\}$, and B is the number of bins on each axis. The descriptor records how scores change over trials within a generation, not an internal latent of the agent. The early bin i is the vertical axis and the late bin j is the horizontal axis, with $i = 0$ at the top, so later improvement lies to the right, and a weaker early score lies toward the top. The grid resolution B is task-specific.

Fitness. To keep scores comparable when different generations write into the same cell, the three fitness terms are scaled by the task-inherent upper bound and are not min-max normalised within a generation:

$$\begin{aligned} f_{\text{perf}} &= \text{clip}(\bar{s}/s_{\max},\, 0, 1), \\ f_{\text{stab}} &= 1/(1+\sigma), \\ f_{\text{div}} &= \text{clip}\big(\text{JS}(p, \bar{p})/\log 2,\, 0, 1\big), \end{aligned} \tag{7}$$

$$f = 0.55\, f_{\text{perf}} + 0.25\, f_{\text{stab}} + 0.20\, f_{\text{div}}. \tag{8}$$

Here, f_{perf}, f_{stab}, and f_{div} denote performance, stability, and diversity, and clip truncates its argument to the stated interval. $\bar{s}$ and σ are the mean and standard deviation of the candidate's K trial scores, p is that candidate's behaviour distribution, and $\bar{p}$ is a reference behaviour distribution. JS denotes the Jensen–Shannon divergence, computed in natural logarithms, with values in $[0, \log 2]$. The distribution p is defined on actions or directed edges depending on the task.

Search. After current-generation candidates are written, search samples a cell uniformly, mutates only that cell's experience c, re-evaluates with π_θ, and attempts to occupy a cell by ϕ and f. Search changes c only, not θ. Mutation operators depend on the data type of c. Experience passed to the next generation is taken only from the weak-early, strong-late region. With $\phi = (i, j)$,

$$\mathcal{R} = \big\{(i, j) : j > i,\ j \geq \lceil (B-1)/2 \rceil \big\}. \tag{9}$$

That is, the late bin is strictly higher than the early bin, and the late bin lies at least in the upper half of the grid. Elites are taken from $\mathcal{A} \cap \mathcal{R}$ (sorted by f, and cycled when fewer are available than needed) and shared as the initial context of every member of the next generation. If the region is empty, CGIRL falls back to the full archive. The next generation therefore receives elite experience from a designated performance region, rather than copies of a single optimum filling every member.

4.3 Training and inference

Training. Training teaches the policy to read a demonstration and then to complete the instance after the demonstration disappears. A demonstrator writes into the observation with probability $P(\mathrm{vis})$, annealed from 1 to 0 by equation 4, so the last trial can stand alone. Training demonstrations need not be optimal. The context in this stage comes from the training demonstrator, gradients are enabled, and only θ is updated. The archive and the search are not used.

Inference. Inference uses the same demonstration channel, now filled by the transmitted experience $c^{(g)}$. Generation $g = 0$ has no previous experience and starts from a task-specific seed. Generation $g > 0$ reads $c^{(g)}$. When the generation ends, $\mathcal{I}$ writes $c^{(g+1)}$ from elites in region $\mathcal{R}$. Larger instances can load a checkpoint trained at a smaller size; only the length-dependent positional embeddings are interpolated. Gradients are off, and only c changes. The procedures are given in Appendix A.

5 Experiments

We evaluate CGIRL on TSP and Memory. Unless stated otherwise, every comparison uses the same trained network and does not train a new network. Only the use of context within a generation, and its transfer between generations, is changed.

5.1 Experimental setup

Tasks. We evaluate CGIRL on TSP and Memory. Memory is a simple hidden-sequence environment: later stages cannot see the sequence except through transferred experience c, so the task is well suited to testing whether cross-generational transfer actually carries what later inference needs (Cook et al., 2024). TSP is a more general and more challenging combinatorial search, suited to testing in-context reasoning. The policy is trained only on TSP50. TSP70/80 load the same checkpoint, so that this ability is probed more fully at scales unseen in training (Ye et al., 2024; Zhou et al., 2025; 2026). We do not run a scale transfer on Memory.

Instance generation. TSP instances sample city coordinates uniformly in the unit square. In-distribution evaluation uses $n = 50$. Scale transfer uses the same TSP50 checkpoint at $n = 70$ and $n = 80$, interpolating only the path positional embeddings. Memory reproduces a fixed hidden sequence. Sequence type and length follow the environment, and we do not run a scale transfer.

Model and training. The TSP policy is an encoder–decoder pointer network (Vinyals et al., 2015; Zhou et al., 2024; Berto et al., 2024; Drakulic et al., 2025; Huang et al., 2025; Li et al., 2025) with embedding size 128, a 6-layer encoder with 8 heads, trained on TSP50 with POMO (Kwon et al., 2020). Training demonstrations are greedy nearest-neighbour tours, with visibility annealed from 1 to 0 over training steps. The Memory policy is a GRU actor-critic trained with PPO (Schulman et al., 2017). Demonstrations are sampled by accuracy, and visibility follows equation 4. Neither environment maintains an archive during training.

Metrics. TSP reports tour length (lower is better); Memory reports recall return (higher is better). Time is per-instance wall-clock time. At test time, parameters need not be updated. The default setting uses $G = 7$ generations and $K = 4$ trials each. TSP uses a parallel population of 5, and Memory uses 5 agents and 5 seeds. Within-generation demonstration visibility decreases by equation 4: the demonstration is always visible on the first trial and never visible on the last. Scores are therefore taken from this last unprompted trial, unless we change G or align the window. The context window length is fixed at 4, matching the default K.

5.2 Main results

We compare five transfer operators. DT uses only the within-generation window and does not write $c^{(g+1)}$. AD concatenates previous states into the next generation. RA-DT retrieves fragments within the current instance by similarity. Direct passes only the current best. CGIRL uses the transfer operator of Section 4 and writes $c^{(g+1)}$ from the region $\mathcal{R}$ when a generation ends. The default is $G = 7$, $K = 4$, $T = 28$ trials, and window $W = 4$. TSP70/80 load the TSP50 checkpoint and were not trained at those sizes. Figure 3 shows the generation-wise process.

Table 1 reports the last unprompted trial. CGIRL has the best terminal score on all six tasks. On TSP50, the tour falls from DT's 6.30 to 5.40 (14% shorter). The same checkpoint shortens unseen TSP70 by 15% and TSP80 by 11%. In the figure, DT is nearly flat while CGIRL decreases over generations, so the gain comes from experience written across generations. On Memory-24, DT is close to failing recall (0.97) while CGIRL reaches 11.11. On Memory-48 and Memory-72, the scores are 21.47 and 33.80. On TSP80, AD is worse than DT: unselected history fills the channel. The improvement is therefore not a tour-specific inductive bias: the same trend holds under scale transfer and on sequence recall.

Table 1: Terminal scores of five transfer operators. TSP reports tour length (lower is better). TSP70/80 load the TSP50 checkpoint. Memory reports recall return, and columns are sequence length. Gap is CGIRL versus DT.

	TSP			Memory		
Method	50	70	80	24	48	72
DT	6.30	7.45	7.66	0.97	4.20	3.33
AD	5.80	6.98	7.89	3.80	3.07	7.67
RA-DT	5.58	6.90	7.42	7.80	14.00	18.00
Direct	5.60	6.82	7.34	8.90	20.73	30.20
CGIRL	**5.40**	**6.36**	**6.82**	**11.11**	**21.47**	**33.80**
Gap	**14%**	**15%**	**11%**	**1045%**	**411%**	**915%**

5.3 Inference cost

We compare cross-generational inference with a single long window under a matched trial budget $T = G \cdot K$. DT places every trial in one generation and sets the window to T. Direct splits the same budget across generations and keeps the window at $W = 4$. At $T = 8$, DT uses G=1, K=8, W=8 and Direct uses G=2, K=4, W=4. At $T = 28$, DT uses G=1, K=28, W=28 and Direct uses G=7, K=4, W=4. Table 2 reports per-instance wall-clock time. Direct is 11%–16% faster at T=8 and about 50% faster at T=28. The gap widens with T. The single-generation window grows with the history, whereas crossing generations keeps the window fixed.

5.4 What to transmit

To isolate the transfer operator $\mathcal{I}$, we keep the rest of CGIRL fixed: Direct copies the current best, and CGIRL archives by descriptor and takes elites from the region $\mathcal{R}$. Figure 4 shows these two curves. CGIRL is shorter on TSP and higher on Memory: 5.40 versus Direct's 5.60 on TSP50, and 11.11 versus 8.90 on Memory. If only the current best is passed on, the population tends to collapse

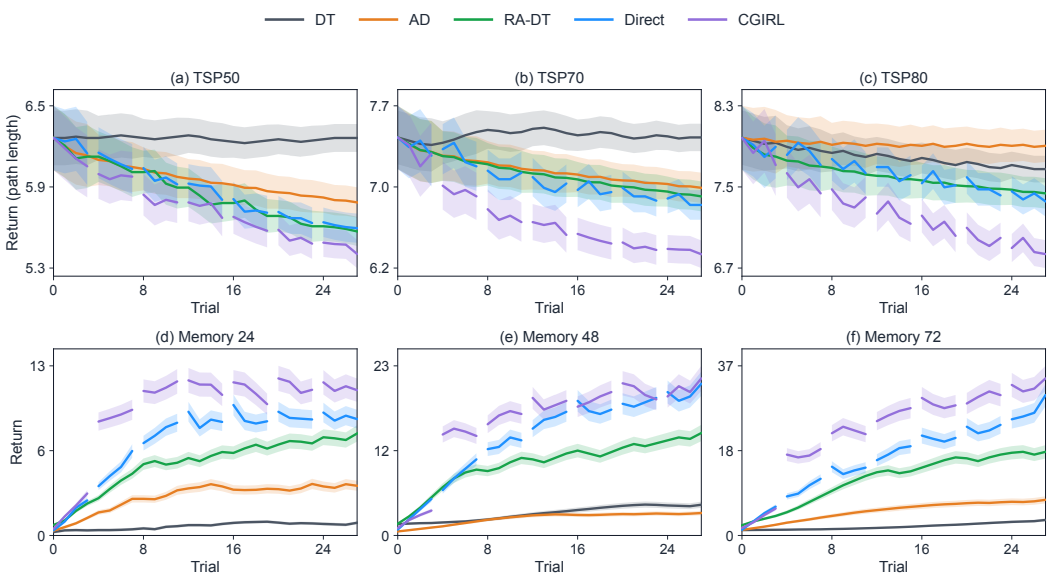

Figure 3: Same trained policy: only the context operator changes. Top: TSP50/70/80 (tour length, lower is better). Bottom: Memory sequence lengths 24/48/72 (return, higher is better). TSP70/80 load the TSP50 checkpoint and were not trained at those sizes. Terminal scores are in Table 1.

Table 2: Per-instance wall-clock time. G is the number of generations, K the trials per generation, W the context window, and $T = G \cdot K$ the total trials. Single generation: DT with $W{=}T$. Cross-generational: Direct with $W{=}4$. Reduction is relative to the matched single-generation window.

Problem	$T{=}8$			$T{=}28$		
	long	short	Red.	long	short	Red.
	$G{=}1$	$G{=}2$		$G{=}1$	$G{=}7$	
	$K{=}8, W{=}8$	$K{=}4, W{=}4$		$K{=}28, W{=}28$	$K{=}4, W{=}4$	
TSP50	0.65s	**0.58s**	**10.8%**	2.36s	**1.24s**	**47.5%**
TSP70	0.87s	**0.74s**	**14.9%**	3.25s	**1.55s**	**52.3%**
TSP80	1.01s	**0.85s**	**15.8%**	3.68s	**1.81s**	**50.8%**

onto a single behaviour. Elites from the weak-early, strong-late region better transmit experience that improved within a generation.

5.5 Case study on transfer channel

We keep the rest of CGIRL fixed and change only the concatenation channel: observation embedding, action, or reward. The observation embedding is computed on the current instance by the encoder and matches the channel used to read demonstrations in training. An action is a discrete choice and drops geometry. A reward is a scalar that only indicates whether a trajectory is good. Table 3 reports terminal scores. The observation embedding is shortest at all three TSP sizes and highest on Memory. On TSP, action is next, and reward is worst. On Memory, reward outperforms action. The default concatenates the observation embedding.

5.6 Case study on context size

This experiment tests how a matched trial budget should be split between context-window length and the number of generations. We fix $W = 4$ and $T = 28$, and compare $(G, K) = (7, 4)$, $(4, 7)$, $(1, 28)$, and $(14, 2)$. The policy still reads only the most recent 4 histories. Table 4 reports terminal

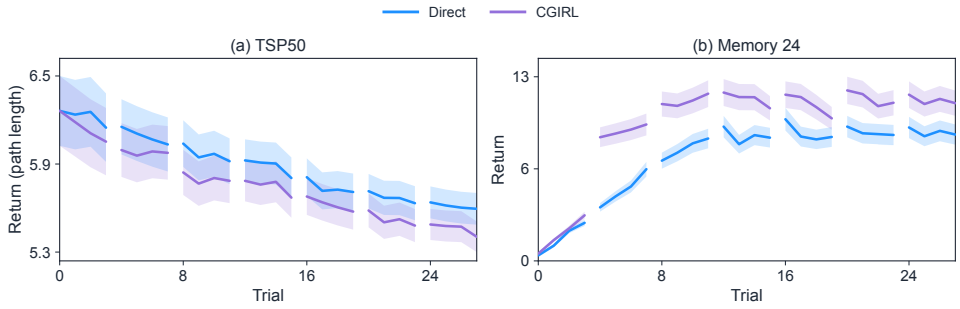

Figure 4: What to transmit: both runs are cross-generational, and only the transfer operator changes. Direct passes the current best. CGIRL archives by descriptor and takes elites from the region $\mathcal{R}$. The TSP y-axis is tour length. The Memory y-axis is return.

Table 3: Transfer channel. The rest of the run is the default CGIRL setting. TSP reports tour length (lower is better). Memory reports recall return (higher is better).

	TSP			Memory
Channel	50	70	80	
Observation	**5.42**	**6.40**	**7.86**	**12.17**
Action	5.50	6.53	8.19	6.33
Reward	5.60	6.71	8.94	10.20

scores. $(7, 4)$ is shortest at all three TSP sizes: relative to a single generation $(1, 28)$, 6.5% shorter on TSP50, 10.2% on TSP70, and 12.0% on TSP80. On Memory, $(7, 4)$ has the highest return (12.17), 123% above $(1, 28)$. $(4, 7)$ lies in between. $(14, 2)$ is substantially worse on Memory, where each generation is too short. A single generation that contains all steps is not equivalent to a longer window. We take $G = 7$, $K = 4$ as the default.

Table 4: Equal-budget split of context across generations. The context window stays 4, and total trials $T = G \cdot K = 28$ under CGIRL. TSP reports tour length (lower is better). Memory reports recall return (higher is better). Parentheses: improvement versus $(1, 28)$.

	TSP			Memory
(G, K)	50	70	80	
$(7, 4)$	**5.42** (6.5%)	**6.40** (10.2%)	**6.76** (12.0%)	**12.17** (123%)
$(4, 7)$	5.62 (3.1%)	6.50 (8.8%)	7.16 (6.9%)	6.27 (15%)
$(14, 2)$	5.72 (1.3%)	6.80 (4.6%)	7.45 (3.0%)	1.60
$(1, 28)$	5.80	7.12	7.68	5.47

6 Conclusion

This paper investigates what a short window should transmit between generations, rather than how to keep lengthening the window. We propose CGIRL, which keeps the window fixed within a generation and adds an evolutionary algorithm, MAP-Elites, to select the experience that crosses generations. Experimental results demonstrate that this cross-generational transfer outperforms a within-generation window on TSP and Memory, and that this evolutionary algorithm further improves on copying only the current best. The transmitted experience is chosen by how returns change within a generation, not by a single terminal score. The window stays fixed, so later generations can keep improving without a longer context.

AI USE STATEMENT

In this work, we used generative AI tools to polish the writing. We have not used generative AI tools to generate synthetic datasets, develop the method or its conceptual framework, formulate or prove mathematical claims, propose hypotheses, design the experiments, implement the methods, clean datasets, conduct qualitative analysis, or interpret the experimental results. Additionally, we used generative AI tools to draft and revise portions of the manuscript, to edit wording for readability, and to identify and format some of the references. We have reviewed all AI-assisted work. The authors checked the revised text, the translations, and the added citations against the source manuscript and the experimental results. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work evaluates algorithms on the travelling salesperson problem and a synthetic sequence-recall task. It does not involve human subjects, personal data, or a deployed system, and it does not release a new dataset. The authors have read the ICLR Code of Ethics and adhered to it.

REPRODUCIBILITY STATEMENT

The model, training setup, evaluation protocol, and default generation and window settings are given in Section 5.1. Reported scores are taken from the last unprompted trial, as stated in Section 5.2. The training and cross-generational inference procedures are given in Appendix A. Task-specific inputs, archives, search, and baseline definitions are given in Appendix B.

REFERENCES

Federico Berto, Chuanbo Hua, Nayeli Gast Zepeda, André Hottung, Niels Wouda, Leon Lan, Junyoung Park, Kevin Tierney, and Jinkyoo Park. RouteFinder: Towards foundation models for vehicle routing problems. *arXiv preprint arXiv:2406.15007*, 2024.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Jonathan Cook, Chris Lu, Edward Hughes, Joel Z. Leibo, and Jakob Foerster. Artificial generational intelligence: Cultural accumulation in reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Juncheng Dong, Moyang Guo, Ethan X. Fang, Zhuoran Yang, and Vahid Tarokh. In-context reinforcement learning from suboptimal historical data. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 14021–14039, 2025.

Darko Drakulic, Sofia Michel, and Jean-Marc Andreoli. GOAL: A generalist combinatorial optimization agent learner. In *International Conference on Learning Representations*, 2025.

Yan Duan, John Schulman, Xi Chen, Peter L. Bartlett, Ilya Sutskever, and Pieter Abbeel. RL^2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

Maxence Faldor and Antoine Cully. Scaling policy gradient quality-diversity with massive parallelization via behavioral variations. *arXiv preprint arXiv:2501.18723*, 2025.

Maxence Faldor, Félix Chalumeau, Manon Flageat, and Antoine Cully. Synergizing quality-diversity with descriptor-conditioned reinforcement learning. *ACM Transactions on Evolutionary Learning and Optimization*, 2024. doi: 10.1145/3696426.

Jake Grigsby, Linxi Fan, and Yuke Zhu. AMAGO: Scalable in-context reinforcement learning for adaptive agents. In *International Conference on Learning Representations*, 2024a. URL https://openreview.net/forum?id=M6XWoEdmwf.

Jake Grigsby, Justin Sasek, Samyak Parajuli, Daniel Adebi, Amy Zhang, and Yuke Zhu. AMAGO-2: Breaking the multi-task barrier in meta-reinforcement learning with transformers. In *Advances in Neural Information Processing Systems*, 2024b. URL https://openreview.net/forum?id=OSHaRf4TVU.

Sili Huang, Jifeng Hu, Zhejian Yang, Liwei Yang, Tao Luo, Hechang Chen, Lichao Sun, and Bo Yang. Decision mamba: Reinforcement learning via hybrid selective sequence modeling. In *Advances in Neural Information Processing Systems*, 2024.

Ziwei Huang, Jianan Zhou, Zhiguang Cao, and Yixin Xu. Rethinking light decoder-based solvers for vehicle routing problems. In *International Conference on Learning Representations*, 2025.

Yeong-Dae Kwon, Jinho Choo, Byoungjip Kim, Iljoo Yoon, Youngjune Gwon, and Seungjai Min. POMO: Policy optimization with multiple optima for reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Michael Laskin, Luyu Wang, Junhyuk Oh, Emilio Parisotto, Stephen Spencer, Richie Steigerwald, DJ Strouse, Steven Hansen, Angelos Filos, Ethan Brooks, Maxime Gazeau, Himanshu Sahni, Satinder Singh, and Volodymyr Mnih. In-context reinforcement learning with algorithm distillation. In *International Conference on Learning Representations*, 2023.

Jonathan N. Lee, Annie Xie, Aldo Pacchiano, Yash Chandak, Chelsea Finn, Ofir Nachum, and Emma Brunskill. Supervised pretraining can learn in-context reinforcement learning. In *Advances in Neural Information Processing Systems*, 2023.

Qi Li, Zhiguang Cao, Yining Ma, Yaoxin Wu, and Yue-Jiao Gong. Diversity optimization for travelling salesman problem via deep reinforcement learning. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2025.

Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909*, 2015.

Alexander Nikulin, Ilya Zisman, Alexey Zemtsov, and Vladislav Kurenkov. XLand-100B: A large-scale multi-task dataset for in-context reinforcement learning. *arXiv preprint arXiv:2406.08973*, 2024.

Thomas Schmied, Fabian Paischer, Vihang Patil, Markus Hofmarcher, Razvan Pascanu, and Sepp Hochreiter. Retrieval-augmented decision transformer: External memory for in-context reinforcement learning. In *Conference on Lifelong Learning Agents*, 2025.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Jaehyeon Son, Soochan Lee, and Gunhee Kim. Distilling reinforcement learning algorithms for in-context model-based planning. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=BfUugGfBE5.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Advances in Neural Information Processing Systems*, volume 28, 2015.

Fan Wang, Pengtao Shao, Yiming Zhang, Bo Yu, Shaoshan Liu, Ning Ding, Yang Cao, Yu Kang, and Haifeng Wang. Towards large-scale in-context reinforcement learning by meta-training in randomized worlds. In *Advances in Neural Information Processing Systems*, 2025.

Jane X. Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z. Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matthew Botvinick. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016.

Haoran Ye, Jiarui Wang, Helan Liang, Zhiguang Cao, Yong Li, and Fanzhang Li. GLOP: Learning global partition and local construction for solving large-scale routing problems in real-time. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 20284–20292, 2024.

Changliang Zhou, Xi Lin, Zhenkun Wang, Xialiang Tong, Mingxuan Yuan, and Qingfu Zhang. Instance-conditioned adaptation for large-scale generalization of neural routing solver. *IEEE Transactions on Intelligent Transportation Systems*, 2026.

Jianan Zhou, Zhiguang Cao, Yaoxin Wu, Wen Song, Yining Ma, Jie Zhang, and Xu Chi. MVMoE: Multi-task vehicle routing solver with mixture-of-experts. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 61804–61824, 2024.

Shipei Zhou, Yuandong Ding, Chi Zhang, Zhiguang Cao, and Yan Jin. DualOpt: A dual divide-and-optimize algorithm for the large-scale traveling salesman problem. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025.

Ilya Zisman, Vladislav Kurenkov, Alexander Nikulin, Viacheslav Sinii, and Sergey Kolesnikov. Emergence of in-context reinforcement learning from noise distillation. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 62832–62846, 2024. URL `https://proceedings.mlr.press/v235/zisman24a.html`.

Appendix of paper "Selecting What to Transmit: Cross-Generational In-Context Reinforcement Learning"

A Training and inference algorithms

This appendix states how CGIRL is trained and how it runs at inference time. Training teaches a policy to read a demonstration and still complete the instance after the demonstration is removed. At inference time, the trained parameters stay fixed, and each generation replaces only the experience passed to the next generation.

In the procedures below, θ denotes the policy parameters, c the experience passed between generations, and c_{train} the demonstration used in training. $P(\text{vis})$ is the probability that this demonstration is visible on a trial. K is the number of trials in one generation, and $\mathcal{A}$ is the archive. $\mathcal{I}$ is the operator that writes the next experience. $\mathcal{R}$ is the weak-early, strong-late region from which that experience is taken.

Algorithm 1 Training: learn to read context

Initialise the policy parameters θ
for each training step **do**
  Sample an instance x and clear the trajectory buffer
  for trial $k = 0, \dots, K-1$ **do**
    $P(\text{vis}) \leftarrow (K-k-1)/(K-1)$
    while the trial is not done **do**
      Observe o, and include the demonstration in o with probability $P(\text{vis})$
      Draw $a \sim \pi_\theta(\cdot \mid x,\, c_{\text{train}})$ and append the step to the buffer
  Update θ from the collected trajectories

Algorithm 2 Cross-generational inference

Load the trained parameters θ, keep gradients off, and do not update θ
for each test instance x **do**
  Initialise $c^{(0)}$ from a task-specific seed, and start from an empty archive $\mathcal{A}$
  for generation $g = 0, 1, \dots$ **do**
    for each parallel agent sharing the same $c^{(g)}$ **do**
      for trial $k = 0, \dots, K-1$ **do**
        $P(\text{vis}) \leftarrow (K-k-1)/(K-1)$
        Include $c^{(g)}$ in the observation with probability $P(\text{vis})$
        Unroll $\tau_k^{(g)} \sim \pi_\theta(\cdot \mid x,\, c^{(g)})$ and append its score to $s^{(g)}$
      Record the last unprompted trajectory and $s^{(g)}$ as the candidate
    For each candidate, set $\phi = \big(\text{bin}(s_{\text{early}}),\, \text{bin}(s_{\text{late}})\big)$ and compute f by equation 8
    Write (c, f) into cell ϕ of $\mathcal{A}$, replacing the occupant when f is higher
    Sample one cell of $\mathcal{A}$ uniformly, and mutate only the c stored in that cell
    Re-evaluate the mutated experience with π_θ, then insert it by ϕ and f
    if $\mathcal{A} \cap \mathcal{R} \neq \emptyset$ **then**
      Take elites from $\mathcal{A} \cap \mathcal{R}$, sorted by f, cycling when fewer are available than needed
    else
      Take elites from the full archive $\mathcal{A}$, sorted by f, and cycle them in the same way
    Share the selected elites as $c^{(g+1)}$ among all members of the next generation

B Environment instantiations

This appendix instantiates CGIRL on Memory and on TSP, which use different data formats. Each specifies what is passed to the next generation, how behavioural diversity is measured, and how search mutates a candidate.

B.1 MEMORY

The task is to reproduce a fixed hidden sequence with a GRU actor-critic trained with PPO. Each observation concatenates the agent's own previous action, a trial-termination flag, the immediate reward, the demonstration action, and a visibility flag. The GRU reads these concatenated fields jointly.

During training, demonstrations are sampled by accuracy: the correct answer, otherwise random. These demonstrations are not experiences left by a previous test-time generation, and the archive is not updated. At test time, both the demonstration and the current generation use the trained parameters, which are not updated.

For Memory, the transmitted experience is a pair (h, o), where h is the GRU hidden state and o is the observation. On trial k, the demonstration is visible with probability $(K - k - 1)/(K - 1)$. It is always visible on the first trial and never visible on the last. We extract (h, o) at the end of the second-to-last trial. This pair is the memory at the start of the final unprompted evaluation.

The first generation starts from a previous-generation state that is randomly initialised. On write, the (h, o) of all parallel agents in the current generation, together with their K-trial score trajectories, are candidates. The behaviour distribution is the policy's one-step action distribution at the snapshot, $p(a) = \pi_\theta(a \mid h, o)$, where a is an action. Search adds Gaussian noise, scaled to the coordinate magnitudes, to (h, o). For $B = 4$, the region $\mathcal{R}$ is $\{(0, 2), (0, 3), (1, 2), (1, 3), (2, 3)\}$. Elites taken from $\mathcal{A} \cap \mathcal{R}$ are the shared initial (h, o) of every member of the next generation.

Direct takes the (h, o) of the best individual by last-trial score and shares it across all members. DT does not condition its policy on the pair (h, o). AD concatenates the current GRU hidden state onto the start of the next generation. RA-DT writes (h, o) produced in the current evaluation into a fixed-capacity FIFO bank. The next generation's demonstration is taken from the entries whose observations are closest in Euclidean distance to the current best individual.

B.2 TSP

The previous generation's tour is not fed in full to cross-attention. The environment proposes the next city along that tour at the current step. The model mean-pools the encodings of those suggested nodes with a mask to obtain $\mathrm{teacher_emb}$. When the demonstration is marked invisible, the corresponding index is -1 and the node is excluded from the pool. Visibility on trial k is $(K - k - 1)/(K - 1)$. Completed-trial historical tours and their returns are tokenized as

$$[R \mid v_1, \ \ldots, \ v_n], \tag{10}$$

where R is a return token for that tour, $v_1, \ldots, v_n$ are the n cities on the tour, and path positional embeddings are added. With the node embedding of the current decoding step as the query, cross-attention over all history tokens yields $\mathrm{context_emb}$. The sum in

$$\mathrm{context_emb} + \mathrm{teacher_emb} \tag{11}$$

is concatenated with the current node embedding and passed to the pointer decoder. Training demonstrations are greedy nearest-neighbour tours, and the training loss is the POMO loss. The visibility probability p_{obs} is annealed from 1 to 0 over training steps. At test time, generation $g = 0$ starts without a previous-generation tour. Generation $g > 0$ is given a previous-generation tour, and the demonstration becomes less visible as the trial index increases.

The first generation is seeded with nearest-neighbour tours that have been temperature-perturbed. On write, the tours of parallel candidates in the current generation and their K-trial score trajectories enter the archive. The behaviour distribution is the empirical distribution over directed edges of a tour. For a tour $(v_1, \ldots, v_n)$,

$$p(v_t \to v_{t+1}) = 1/n, \qquad v_{n+1} := v_1, \tag{12}$$

and all other directed edges are zero. On a closed tour, each city is visited once, so city-visit frequencies do not discriminate. Behavioural diversity is defined on the directed edges of a tour. Search applies random segment reversals to the tour. For $B = 3$, the region $\mathcal{R}$ is $\{(0, 1), (0, 2), (1, 2)\}$, and tours taken from $\mathcal{R}$ are shared by the parallel agents. TSP70/80 load a TSP50 checkpoint, and only the path positional embeddings are linearly interpolated to the new length.

Direct keeps only the historically shortest tour and replaces it when the current generation finds a shorter one. DT provides no previous-generation tour and does not retain historical tours across generations. AD accumulates each trial's tour across generations, up to the most recent $G \cdot K$ tours. That cap is set to $7 \times 4 = 28$, matching the total number of trials. RA-DT writes previous tours and subtours of the current instance into a bank, keyed by the mean of encoder node embeddings. Retrieval uses cosine similarity to the current best tour's embedding and reweights candidates by tour length. The top-k retrieved entries are then inserted into the context.