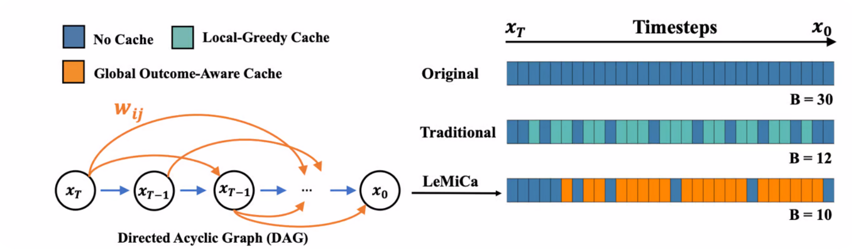
LeMiCa

- □□□□□LeMiCa: Lexicographic Minimax Path Caching for Efficient Diffusion-Based Video Generation
- □□□□□https://arxiv.org/abs/2511.00090
- NIPS 2025



□□□□□□□□□□cache□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□cache□□□t1□t2□□□□□□□□□□□□□□□□□□□□□□

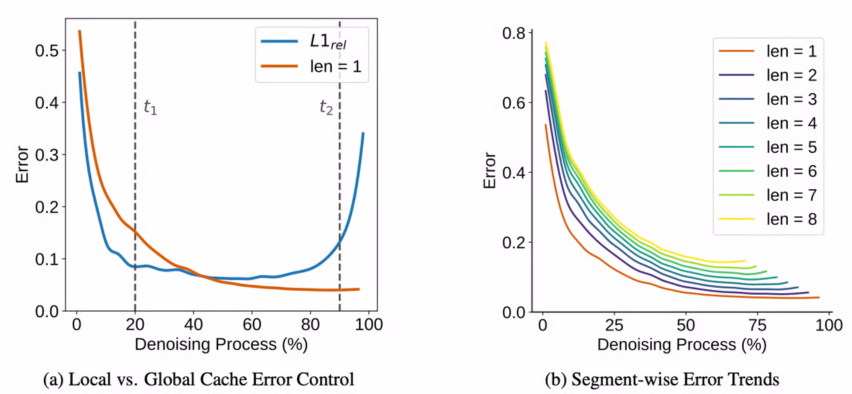


Figure 2: Rethinking cache reuse in denoising diffusion via error estimation. (a) The traditional Local-Greedy ($L1_{rel}$) strategy uses fixed thresholds on local output differences between adjacent timesteps to decide when to cache. This assumes uniform temporal sensitivity, which can be misleading—for instance, caching at t_2 yields lower final error than t_1, despite t_1 seeming smoother locally. This highlights the role of temporal heterogeneity. (b) Our Global Outcome-Aware (*segment-wise error*) strategy estimates final output error when caching outputs over segments of length len, starting from timestep i. The plot shows that early caches cause greater error, supporting an outcome-sensitive, trajectory-aware strategy over fixed local heuristics.

□□a□□□□□□□□cache□□□□□□□□□□□□□cache□□□□□□□□□□□□□□□□error□cache□□□□□□□□□□□□b□

□□□□□□□□□□□□loss□□□□□□□□cache□□□□□□□□□□□□□□

$$\mathrm{L1}_{\mathrm{glob}}(i \to j) = \frac{1}{N} \left\| x_0^{\mathrm{cache}(i \to j)} - x_0^{\mathrm{original}} \right\|_1$$

□□□□□□□□□□□□□□□□□□□□cache□□□□□□□□□□□□□□□□□□□□□□DAG□□□□□□□□□□□i□j□□cache□□□□loss□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□prompt□□

Algorithm 1 Lexicographic Minimax Path Selection

Input: Directed acyclic graph $G = (V, E)$, start node s, end node t, step limit B
Output: Lexicographic Minimax Path P^*
Initialization:
  $dp[v][k]$: maximum edge weight on any k-step path to v
  $paths[v][k], edges[v][k]$: corresponding node and edge sequences
  $dp[s][0] \leftarrow 0, paths[s][0] \leftarrow [[s]], edges[s][0] \leftarrow [[\,]]$
Main Loop:
for $k = 0$ to $B - 1$ **do**
  for each node v with $dp[v][k] < \infty$ **do**
    for each neighbor u of v **do**
      $w \leftarrow$ weight of edge (v, u)
      $m \leftarrow \max(dp[v][k], w)$
      if $m < dp[u][k+1]$ **then**
        $dp[u][k+1] \leftarrow m$
        Update $paths[u][k+1], edges[u][k+1]$ from v
      else if $m = dp[u][k+1]$ **then**

```
            Append new paths and edges from v to paths[u][k + 1], edges[u][k + 1]
        end if
    end for
  end for
end for
Final Selection:
P* ← min(zip(paths[t][B], edges[t][B]), key = λ(p, e) : sorted(e, reverse=True))
```

□□□□□□□□

□□□□□□
1□□□□□□□□□□□□□□□prompt□□□□□□□□□□□□□□prompt□□□□□
2□□□□□□□□□cache□□□□□□□□□□□□□□□□□cache□□□□□segment□cache□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□segment□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

- □□□https://github.com/UnicomAI/LeMiCa
