IMPROVING OPTIMALITY AND SPEED OF GREEDY GROUP RECURSION ALGORITHM

Florin Dobrian [1] **Oleg Puzyrko White** [1]

ABSTRACT

Recently (Liu et al., 2025) has presented efficient algorithm - Greedy Group Recursion (GGR) - for reordering the rows and the fields within each row of an input table to maximize key-value (KV) cache reuse when performing LLM serving. In this paper, we propose several adjustments to GGR algorithm that can improve optimality of the solution and reduce its execution time.

1 INTRODUCTION

There has been growing research on LLM inference optimization. In particular, recent work (Liu et al., 2025; Cheng et al., 2025) presents solutions to optimize relational data analytics workloads for offline LLM inference. It proposes Greedy Group Recursion (GGR), an approximate algorithm that leverages functional dependencies (such as primary and foreign key relationships from the data schema) and table statistics, which are readily available in many databases and analytics systems, to reduce the search space.

The GGR algorithm is described in detail in (Liu et al., 2025). Let us recup briefly its main points.

The pseudocode specification of GGR is in Algorithm 1 and is taken from the original paper but with several critical corrections (and multiple little typo fixes).

Algorithm 1 Greedy Group Recursion (GGR)

Input: Table T, Functional Dependency FD
Output: Prefix Hit Count S, Reordered List of Tuples L

function HITCOUNT(v, c, T, FD)
  $R_v \leftarrow \{i \mid T[i, c] = v\}$
  inferred_cols $\leftarrow \{c' \mid (c, c') \in FD\}$
  inferred_vals $\leftarrow \{T[R_v[1], c'] \mid c' \in$ inferred_cols$\}$
  tot_len $\leftarrow$ len$(v)^2 + \sum_{c' \in \text{inferred_cols}} \left(\frac{\sum_{r \in R_v} \text{len}(T[r, c'])}{|R_v|} \right)^2$
  $HC \leftarrow$ tot_len $\times$ $(|R_v| - 1)$
  $cols \leftarrow [c] +$ inferred_cols
  $vals \leftarrow [v] +$ inferred_vals
  return $HC, cols, vals$
end function

function GGR(T, FD)
  if $|T|_{rows} = 1$ **then**
    return $0, [T[1]]$
  end if
  if $|T|_{cols} = 1$ **then**
    $S \leftarrow \sum_{v \in \text{distinct}(T[,1])}$ HITCOUNT$(v, 1, T)$
    return $S, sort([T[i] \mid i \in 1 \dots |T|_{rows}])$
  end if
  $b\_v, b\_c \leftarrow$ None, None
  $max\_HC, b\_cols, b\_vals \leftarrow -1, [], []$
  for $c \in$ columns(T), $v \in$ distinct$(T[, c])$ **do**
    $HC, cols, vals \leftarrow$ HITCOUNT(v, c, T, FD)
    if $HC > max\_HC$ **then**
      $b\_v, b\_c \leftarrow v, c$
      $max\_HC, b\_cols, b\_vals \leftarrow HC, cols, vals$
    end if
  end for
  $R\_v \leftarrow \{i \mid T[i, b\_c] = b\_v\}$
  $A\_HC, A\_L \leftarrow$ GGR$(T[\text{rows} \setminus R_v, \text{cols}], FD)$
  $B\_HC, B\_L \leftarrow$ GGR$(T[R_v, \text{cols} \setminus b_cols], FD)$
  $C\_HC \leftarrow max\_HC$
  $S \leftarrow A\_HC + B\_HC + C\_HC$
  $L \leftarrow [b\_vals + B\_L[i] \mid i \in 1 \dots |R\_v|] + A\_L$
  return S, L
end function

return GGR(T, FD)

REFERENCES

Cheng, A., Liu, S., Pan, M., Li, Z., Agarwal, S., Cemri, M., Wang, B., Krentsel, A., Xia, T., Park, J., Yang, S., Chen, J., Agrawal, L., Naren, A., Li, S., Ma, R., Desai, A., Xing, J., Sen, K., Zaharia, M., and Stoica, I. Let the barbarians in: How ai can accelerate systems performance research, 2025. URL https://arxiv.org/abs/2512.14806.

Liu, S., Biswal, A., Kamsetty, A., Cheng, A., Schroeder, L. G., Patel, L., Cao, S., Mo, X., Stoica, I., Gonzalez, J. E., and Zaharia, M. Optimizing llm queries in relational data analytics workloads, 2025. URL https://arxiv.org/abs/2403.05821.

[1]Data Analytics Group. Corresponding author: Oleg P. White <oleg.p.white@gmail.com>.