A key concern is the robustness of our partitioning strategy given the potential for inaccuracies in the fitted performance model parameters (a, b, β, γ). The robustness of our framework stems not from the absolute precision of the underlying performance model, but from the dynamic, two-phase decision-making logic of the `FindBestSplit` algorithm throughout its iterative process.

1 Phase 1: The Initial Split as a High-Gain Operation

The first phase involves splitting the largest source partition, π_{src}, which contains a set of roles R_{src}. The algorithm's first move is to take a **single role**, r^*, and place it in a new, empty partition, π_{new}. The remaining roles, $R_{\text{rem}} = R_{\text{src}} \setminus \{r^*\}$, stay in the updated source partition π'_{src}.

Performance Gain Analysis. Before the split, the average query cost for all roles in R_{src} is determined by the single, large, low-selectivity source partition:

$$C_{\text{avg, before}} = \frac{1}{|R_{\text{src}}|} \sum_{r \in R_{\text{src}}} \log\big(|\pi_{\text{src}}|\big) \big(a \cdot ef_{s,\text{before}} + b\big). \tag{1}$$

After the split, the new average cost for the same set of roles is a weighted average of the costs in their new partitions. The cost for the single moved role r^* is now calculated in its own highly efficient partition π_{new}:

$$C_{\text{avg, after}} = \frac{1}{|R_{\text{src}}|} \left[\sum_{r \in R_{\text{rem}}} \log\big(|\pi'_{\text{src}}|\big) \big(a \cdot ef_{s,\text{rem}} + b\big) \; + \; \log\big(|\pi_{\text{new}}|\big) \big(a \cdot ef_{s,\text{split}} + b\big) \right]. \tag{2}$$

The performance gain is driven by the change in average user selectivity, $\overline{s_u}$. Before the split, all roles share a low selectivity, $\overline{s_{u,\text{before}}}$, within the large partition. After the split:

- For the moved role r^*, its new partition π_{new} is highly specialized, causing its associated user selectivity $\overline{s_{u,\text{split}}}$ to increase dramatically.
- For the remaining roles R_{rem}, the removal of r^*'s documents also makes their partition more focused, increasing their average selectivity to $\overline{s_{u,\text{rem}}} > \overline{s_{u,\text{before}}}$.

According to the recall model, the required search depth is inversely proportional to selectivity ($ef_s \propto 1/\overline{s_u}$). Therefore, the sharp increase in selectivity directly leads to a significant decrease in the required search depths:

$$ef_{s,\text{split}} \ll ef_{s,\text{before}} \quad \text{and} \quad ef_{s,\text{rem}} < ef_{s,\text{before}}.$$

This drastic reduction in the ef_s terms, combined with the smaller partition sizes, is what causes the overall average cost $C_{\text{avg, after}}$ to be significantly lower than $C_{\text{avg, before}}$.

Robustness in Phase 1. The resulting performance gain, $\Delta Q = C_{\text{avg, after}} - C_{\text{avg, before}}$, is a large negative number. This decision is driven by a strong, unambiguous signal. It is highly robust and insensitive to minor errors in the model parameters, as the magnitude of the improvement far outweighs any potential modeling noise.

2 Phase 2: Subsequent Splits as Efficiency Optimization

Subsequent iterations involve moving another single role r^* from the shrinking source partition π_{src} to the growing, non-empty destination partition π_{dst}.

Cost-Benefit Analysis. In this phase, the raw performance gain $-\Delta Q$ is typically smaller. The robustness of the algorithm is now derived from its objective function, which optimizes for the best cost-benefit ratio for moving a single role:

$$\max \frac{\Delta Q}{\Delta S}$$

Here, ΔS represents the change in memory overhead (data overlap) from moving the single role r^*, calculated as $\Delta S = (|\pi'_{\text{src}}| + |\pi'_{\text{dst}}|) - (|\pi_{\text{src}}| + |\pi_{\text{dst}}|)$.

Robustness in Phase 2. The algorithm's decision criteria dynamically shift from maximizing raw performance to optimizing the balance between performance and memory efficiency. At each step, it evaluates every possible single-role move and robustly selects the best one:

- A move with a moderate performance gain (ΔQ_A is moderately negative) but a large memory cost (ΔS_A is large) may be ranked poorly.
- Conversely, a move with only a marginal performance gain (ΔQ_B is slightly negative) may be chosen as optimal if it also effectively reduces data overlap (ΔS_B is very small or even 0).

3 Conclusion

The system's design ensures that its partitioning decisions are highly robust to the exact values of the performance parameters (a, b, β, γ), since it prioritizes "high signal-to-noise ratio" operations whose performance gains far outweigh model errors, while in cases of uncertain benefits it mitigates risks by either shifting toward optimizing memory efficiency or safely terminating the iteration.