Efficient Selection of Informative Alternative Relational Query Plans for Database Education

Hu Wang†, Hui Li†, Sourav S. Bhowmick†, Zihao Ma†, Jiangtao Cui†

† School of Computer Science and Engineering, Nanyang Technological University, Singapore

Abstract—Given an SQL query, a relational query engine selects a query execution plan (QEP) for processing it. Off-the-shelf RDBMS typically only expose the selected QEP to users without revealing information about representative alternative query plans (AQP) considered during QEP selection in a user-friendly manner. Easy access to representative AQPs is useful in several applications such as database education where it can facilitate learners to comprehend the QEP selection process during query optimization. In this paper, we present a novel end-to-end and generic framework called ARENA that facilitates exploration of informative alternative query plans of a given SQL query to aid the comprehension of QEP selection. Under the hood, ARENA addresses a novel problem called informative plan selection problem (TIPS) which aims to discover a set of informative alternatives from the underlying plan space so that the plan interestingness of the set is maximized. Specifically, we explore two variants of the problem, namely batch TIPS and incremental TIPS, to cater to diverse set of users. Due to the computational hardness of the problem, we present a 2-approximation algorithm to address it efficiently. Exhaustive experimental study demonstrates the superiority of ARENA to several baseline approaches. As a representative use case of ARENA, we demonstrate its effectiveness in supplementing database education through a user study and academic assessment involving real-world learners.

Index Terms—Query processing and optimization

I. INTRODUCTION

A relational query engine selects a query execution plan (QEP), which represents an execution strategy of an SQL query, from many alternative query plans (AQP) by comparing their estimated costs. Major off-the-shelf relational database systems (RDBMS) typically only expose the QEP to an end user. They do not reveal a representative set of AQPs considered by the underlying query optimizer during the selection of a QEP in a user-friendly manner. Although an RDBMS (e.g., PostgreSQL) may allow one to manually pose SQL queries with various constraints on configuration parameters to view the corresponding QEPs containing specific physical operators (e.g., SET enable_hashjoin = true), such strategy demands not only familiarity of the syntax and semantics of configuration parameters, but also a clear idea of the AQPs that one is interested. This is often impractical to assume (e.g., in education environment where many learners are taking the database systems course for the first time). Consider the following motivating scenario.

Example 1.1: Lena is currently enrolled in an undergraduate database systems course. She wishes to explore some of the representative AQPs for the following query from the TPC-H benchmark after viewing its QEP as depicted in Fig. 1(a).

\[
\begin{align*}
\text{SELECT} & \quad \text{s_acctbal, s_name, n_name, s_address, s_phone} \\
\text{FROM} & \quad \text{supplier, nation, region} \\
\text{WHERE} & \quad \text{s_acctbal} \geq 5000 \text{ and s_nationkey} = n_nationkey \text{ and n_regionkey} = r_regionkey;
\end{align*}
\]

Specificially, Lena is curious to know whether there are AQPs that have similar (resp. different) structure and physical operators but very different (resp. similar) estimated cost? If there are, then how they look like? Examples of such AQPs are shown in Fig. 1(b)-(d).

Clearly, a user-friendly tool that can facilitate retrieval and exploration of “informative” AQPs associated with a given query can greatly aid in answering Lena’s questions. Unfortunately, very scant attention has been given to build such tools by the academic and industrial communities. Recent efforts for technological support to augment learning of relational query processing primarily focused on the natural language descriptions of QEPs [28; 43] and visualization of the plan space [16]. In this paper, we present an end-to-end framework called ARENA (AlTerative quEry plaN ExplorAtion) to address this challenge. Our approach not only has positive feedback from learners (Section VIII-A) but, more importantly, eventually improves academic outcomes (Section VIII-B).

Selecting a set of representative AQPs is a technically challenging problem. Firstly, the plan space can be prohibitively large [6]. Importantly, not all AQPs potentially support understanding of the QEP selection process. Hence, it is ineffective to expose all plans. Instead, it is paramount to select those that are informative. However, how do we quantify informativeness in order to select plans? For instance, in Fig. 1(b)-(d), which plans should be revealed to Lena? Certainly, any informativeness measure needs to consider the plans (including the QEP) that an individual has already viewed for her query so that highly similar plans are not exposed to them. It should also be cognizant of individuals’ interests in this context. Secondly, it can be prohibitively expensive to scan all AQPs to select informative ones. How can we design efficient techniques to select informative plans? Importantly, this cannot be integrated
into the enumeration step of the query optimizer as informative AQP need to be selected based on the QEP an individual has seen.

Thirdly, at first glance, it may seem that we can select $k > 1$ alternative query plans where $k$ is a value specified by a user. Although this is a realistic assumption for many top-$k$ problems, our engagement with users reveal that they may not necessarily be confident to specify the value of $k$ always (detailed in Section VIII-A). One may prefer to iteratively view one plan-at-a-time and only cease exploration once they are satisfied with the understanding of the alternative plan choices made by the optimizer for a specific query. Hence, $k$ may not only be unknown apriori but also the selection of an AQP at each iteration to enhance their understanding of different plan choices depends on the plans viewed by them thus far. This demands for a flexible solution framework that can select alternative query plans in the absence or presence of $k$ value. Lastly, any proposed solution should be generic so that it can be easily realized on top of majority of existing off-the-shelf RDBMS.

In this paper, we formalize the aforementioned challenges into a novel problem called informative plan selection (TIPS) problem and present ARENA to address it. Specifically, given an SQL query and $k \geq 1$ (when $k$ is unspecified it is set to 1), it retrieves $k$ alternative query plans (AQP) that maximize plan informativeness. Such query plans are referred to as informative query plans. To this end, we introduce two variants of TIPS, namely batch TIPS (B-TIPS) and incremental TIPS (I-TIPS), to cater for scenarios where $k$ is specified or unspecified by a user, respectively. We introduce a novel concept called plan informativeness to capture the aforementioned challenges associated with the selection of informative AQPs. Specifically, it embodies the preferences of real-world users (based on their feedback) by mapping them to distance (computed by exploiting differences between plans w.r.t. structure, content and estimated cost) and relevance of alternative plans with respect to the QEP or the set of alternative plans viewed by a user thus far. Next, given the computational hardness of the TIPS problem, we present a 2-approximation algorithm to address both the variants of the problem efficiently. Specifically, it has a linear time complexity w.r.t. the number of candidate AQPs.

Given that the number of such candidates can be very large for some queries, we propose two pruning strategies to prune them efficiently. We adopt ORCA [39] to retrieve the candidate plan space for a given SQL query in ARENA as it enables us to build a generic framework that can easily be realized on majority of existing RDBMS.

Lastly, we demonstrate the usefulness of ARENA by deploying it in the database systems course at Xidian University. Specifically, we analyze the feedback from real-world learners and their academic performance w.r.t. the understanding of the QEP selection process. Our analysis reveals the efficiency and effectiveness of our solutions to help learners understand alternative query plans in the context of QEP selection.

In summary, this paper makes the following key contributions.

- We adopt a learner-centered approach to characterize the notion of informative query plans for database education by engaging real-world learners (Section II), and present the notion of plan informativeness (Section IV) to quantitatively model the informativeness of AQP.
- We present a novel informative plan selection (TIPS) problem to obtain a collection of alternative plans for a given SQL query by maximizing plan informativeness (Section III).
- We present approximate algorithms with quality guarantee that exploit plan informativeness and two pruning strategies to address the TIPS problem efficiently (Section VI). Our algorithms underpin the proposed end-to-end ARENA framework to enable users to retrieve and explore informative plans for their SQL queries (Section V).
- We undertake an exhaustive performance study to demonstrate the superiority of ARENA to representative baselines as well as justify the design choices (Section VII).
- We conduct a user study and academic assessment involving real-world learners taking the database systems course to demonstrate the usefulness and effectiveness of ARENA for pedagogical support to supplement their understanding of the QEP selection process (Section VIII).

II. INFORMATIVE ALTERNATIVE PLANS

A key question that any alternative query plan (alternative plan for brevity) selection technique needs to address is as
TABLE I: The 8 alternative plan categories.

| Plan category | Structure | Content | Cost |
|---------------|-----------|---------|------|
| $AP_I$        | small     | small   | small|
| $AP_{II}$     | small     | small   | large|
| $AP_{II}$     | small     | large   | small|
| $AP_{IV}$     | large     | small   | small|
| $AP_V$        | small     | large   | large|
| $AP_{IV}$     | large     | small   | large|
| $AP_{V_{II}}$ | large     | large   | small|
| $AP_{V_{III}}$| large     | large   | large|

![Survey results](image)

follows: What types of alternative plans are “interesting” or “informative”? Reconsider Example 1.1. Fig. 1(b)-(d) depict three alternative plans for the query where the physical operator/join order differences are highlighted with red rectangles and significant cost differences are shown using orange nodes. Specifically, all three AQP’s have different join orders w.r.t. the QEP. However, $AP_I$ has very similar structure as the QEP but significantly higher estimated cost. $AP_2$, on the other hand, has a very similar estimated cost as the QEP. $AP_3$ consumes similar high cost as $AP_I$. Suppose $k = 2$. Then, among $\{AP_I, AP_2\}$, $\{AP_I, AP_3\}$, and $\{AP_2, AP_3\}$, which one is the best choice? Intuitively, it is reasonable to choose the set that is most “informative” to a learner.

For example, by viewing $AP_I$ or $AP_3$, Lena learned that the HASH JOIN operator is sensitive to the order of the joined tables. This reinforces the knowledge she has acquired from her classroom lectures. On the other hand, $AP_2$ reveals that changing the join order may not always lead to significantly different estimated costs. Lena was not aware of this and therefore found it informative. Hence, $\{AP_I, AP_2\}$ or $\{AP_2, AP_3\}$ are potentially the most informative set for Lena.

**Feedback from volunteers.** The above example illustrates that the content, structure, and cost of a query plan w.r.t. the QEP and already-viewed plans play pivotal roles in determining informativeness of AQP’s. To further validate this, we survey 68 unpaid volunteers who have taken a database systems course recently or currently taking it in an academic institution. Specifically, they are either current undergraduate students in a computer science degree program or junior database engineers who have recently joined industry. Note that we focus on this group of volunteers as a key application of ARENA is in database education.

We choose AQP’s of two SQL queries on the IMDb dataset for the volunteers to provide feedback. These queries contain 3-4 joins and involve different table scan methods. Note that randomly selecting a large number of alternative plans for feedback is ineffective as they not only may insufficiently cover diverse types of plans w.r.t. the QEPs of the queries but also our engagement with the volunteers reveal that it may deter them to give feedback as they have to browse a large number of plans. Hence, for each query, we display the QEP and 8 alternative plans, which are heuristically selected by evaluating their differences w.r.t. the QEP based on three dimensions, i.e., structure, content, and estimated cost (Table I). For instance, we may choose an alternative plan with small difference in structure or content from the QEP but large differences in the other two dimensions. Observe that it covers majority of the AQP’s of a given query. Note that the heuristic model may occasionally select some similar alternative plans. We deliberately expose these plans to the volunteers to garner feedback on similar alternative plans.

We ask the volunteers to choose the most and least informative plans given a QEP and provide justifications for their choices. The aforementioned plan types were not revealed to the volunteers. We received 132 feedbacks from the volunteers. We map the feedbacks on informative and uninformative plans to the plan types in Table I and count the number of occurrences. Fig. 2(a) reports the feedback on plans considered as informative. Specifically, $G_1$ represents $AP_{II}$, $G_2$ and $G_3$ represent plans with lower cost differences. Particularly, $G_2$ are plans with large structure and content differences ($AP_{IV}$) whereas $G_3$ are those that have small structural or content differences ($AP_{V_{II}I}$, $AP_{IV}$). $G_1$ represents $AP_{IV}$ and $AP_{V_{III}}$. There are other feedbacks that are too sparse to be grouped as representative of the volunteers.

Fig. 2(b) reports the feedback on uninformative plans selected by the volunteers. $B_1$ represents plans that are very similar to the QEP ($AP_{I}$) and hence voted as uninformative as they do not convey any useful information. $B_2$ represents $AP_{IV}$ and $AP_{V_{III}}$. $B_3$ are plans that are very similar to other alternative plans viewed by the volunteers. In addition, there are eight other feedbacks that are orthogonal to our problem (i.e., feedback on the visual interface design) and hence categorized as “irrelevant”.

**Summary.** Our engagement with volunteers reveal that majority find alternative plans of types $AP_{II} - AP_{IV}$ and $AP_{V_{III}}$ informative. However, $AP_{I}$, $AP_{IV}$, $AP_{II}$, and $AP_{V_{III}}$ are not. Observe that the number of feedbacks for the latter two that deem them uninformative is significantly more than those that consider them informative ($G_1$ vs $B_2$). Hence, we consider them uninformative. Volunteers also find similar alternative plans uninformative. In the sequel, we shall use these insights to guide the design of plan informativeness and pruning strategies.

III. INFORMATIVE PLAN SELECTION (TIPS) PROBLEM

In this section, we formally define the TIPS problem. We begin with a brief background on relational query plans.

A. Relational Query Plans

Given an arbitrary SQL query, an RDBMS generates a query execution plan (QEP) to execute it. A QEP consists of a collection of physical operators organized in form of a tree, namely the physical operator tree (operator tree for brevity). Fig. 1(a) depicts an example QEP. Each physical operator,
B. Problem Definition

The informative plan selection (TIPS) problem aims to automatically select a small number of alternative plans that maximize plan informativeness. These plans are referred to as informative alternative plans (informative plans/AQPs for brevity). The TIPS problem has two flavors:

- Batch TIPS. A user may wish to view top-k informative plans beside the QEP;
- Incremental TIPS. A user may iteratively view an informative plan besides what have been already shown to him/her.

Formally, we define these two variants of TIPS as follows.

**Definition 3.1:** Given a QEP $\pi^*$ and a budget $k$, the batched informative plan selection (B-TIPS) problem aims to find top-k alternative plans $\Pi_{OPT}$ such that the plan informativeness of the plans in $\Pi_{OPT} \cup \{\pi^*\}$ is maximized, that is

$$\Pi_{OPT} = \text{argmax}_{\Pi} U(\Pi \cup \{\pi^*\}) \text{ s.t. } |\Pi| = k \text{ and } \Pi \subseteq \Pi^* \setminus \{\pi^*\}$$

where $\Pi^*$ denotes the space of all possible plans and $U(\cdot)$ denotes the plan informativeness of a set of query plans.

**Definition 3.2:** Given a query plan set $\Pi$ a user has seen so far, the iterative informative plan selection (1-TIPS) problem aims to select an alternative plan $\pi_{OPT}$ such that the plan informativeness of the plans in $\Pi \cup \{\pi_{OPT}\}$ is maximized, that is

$$\pi_{OPT} = \text{argmax}_{\pi} U(\Pi \cup \{\pi\}) \text{ subject to } \pi \in \Pi^* \text{ and } \pi \notin \Pi$$

In the next section, we shall elaborate on the notion of plan informativeness $U(\cdot)$ to capture informative plans.

**Example 3.3:** Reconsider Example 1.1. Suppose Lena has viewed the QEP in Fig. 1(a). She may now wish to view another two alternative plans (i.e., $k = 2$) that are informative. The B-TIPS problem is designed to address it by selecting these alternative plans from the entire plan space so that the plan informativeness $U(\cdot)$ of the QEP and the selected plans is maximized. On the other hand, suppose another learner is unsure about how many alternative plans should be viewed or is dissatisfied with the result returned by B-TIPS. In this case, the 1-TIPS problem enables him to iteratively select “AQP-at-a-time” and provide a rating w.r.t. its informativeness until satisfied. The plan informativeness is maximized by considering the rating from the learner.

**Theorem 3.4:** Given a plan informativeness measure $U(\cdot)$ and the plan space $\Pi^*$, we can find an alternative plan for the 1-TIPS problem in $O(|\Pi^*|)$ time. However, plan space increases exponentially with the increase in the number of joins and it is NP-hard to find the best join order [20]. Therefore, we need efficient solutions especially when there is a large number of joined tables.

**Proof:** (Sketch). Let us consider the facility dispersion problem which is NP-hard [35]. In that problem, one is given a facility set $S$, a function $f$ of distance and an integer $k(k < |F|)$. The objective is to find a subset $S'(|S'| = k)$ of $S$ so that the given function on $f$ is maximum. Because facility dispersion problem is a special case of B-TIPS where $\{\pi^*\} = \emptyset$ and $U(\cdot) = f$, B-TIPS is also NP-hard.

IV. Quantifying Plan Interestingness

In this section, we describe the quantification of plan informativeness $U(\cdot)$ introduced in the preceding section by exploiting the insights from learner-centered characterization of AQPs in Section II.

A. Lessons from Characterization of AQPs

In Section II, feedbacks from volunteers reveal that the selection of informative plans is influenced by the followings:

(a) The structure, content, and cost differences of an alternative plan w.r.t. the QEP (and other alternative plans) can be exploited to differentiate between informative and uninformative alternative plans. Hence, we need to quantify these differences for informative plan selection. (b) Volunteers typically do not prefer to view alternative plans that are very similar to the QEP or other alternative plans already viewed by them. We now elaborate on the quantification approach to capture two factors and then utilizing them to quantify plan interestingness.

B. Plan Differences

Broadly, $U(\cdot)$ should facilitate identifying specific alternative plans other than the QEP such that by putting them together a user can acquire a comprehensive understanding of
the alternative plan choices made by a query optimizer for her query (i.e., informative). This entails quantifying effectively differences between different plans w.r.t. their structure, content, and estimated cost. As $U(\cdot)$ is defined over a set of query plans, we shall first discuss the approach of quantifying these differences.

**Structural Difference.** To evaluate the structural difference between a pair of physical operator trees, we need to employ a measure to compute structural differences between trees. The choice of this measure is orthogonal to our framework and any distance measure that can compute the structural differences between trees efficiently can be adopted. By default, we use the subtree kernel, which has been widely adopted for tree-structured data [38] to measure the structural difference.

Formally, subtree kernel is defined as follows.

$$\kappa_S(T_a, T_b) = \sum_{n_a \in N(T_a)} \sum_{n_b \in N(T_b)} \Delta(n_a, n_b)$$

where $\Delta(n_a, n_b) = \sum_{I(\cdot)} I(n_a)I(n_b)$, and where $I(\cdot)$ is an indicator function which determines whether the proper subtree $f_i$ is rooted at node $n$ and $N(T)$ denotes the node set of $T$.

For example, the kernel score in Fig. 3 is $\kappa_S(T_1, T_2) = 1 \times 1 + 1 \times 1 = 2$ because they share the substructures $f_1$, $f_2$. Since the subtree kernel is a similarity measure, we need to convert it to a distance. Moreover, each dimension may have different scale. Hence, it is necessary to normalize each dimension distance. For subtree kernel $\kappa(T_a, T_b)$, we first compute the normalized kernel as follows and then define the structural distance:

$$\kappa(T_a, T_b) = \kappa_S(T_a, T_b) / \sqrt{\kappa_S(T_a, T_a) \cdot \kappa_S(T_b, T_b)}$$

**Definition 4.2:** Given two trees $T_a$ and $T_b$, the subtree kernel is:

$$\kappa_S(T_a, T_b) = \sum_{n_a \in N(T_a)} \sum_{n_b \in N(T_b)} \Delta(n_a, n_b)$$

where $\Delta(n_a, n_b) = \sum_{I(\cdot)} I(n_a)I(n_b)$, and where $I(\cdot)$ is an indicator function which determines whether the proper subtree $f_i$ is rooted at node $n$ and $N(T)$ denotes the node set of $T$.

For example, the kernel score in Fig. 3 is $\kappa_S(T_1, T_2) = 1 \times 1 + 1 \times 1 = 2$ because they share the substructures $f_1$, $f_2$. Since the subtree kernel is a similarity measure, we need to convert it to a distance. Moreover, each dimension may have different scale. Hence, it is necessary to normalize each dimension distance. For subtree kernel $\kappa(T_a, T_b)$, we first compute the normalized kernel as follows and then define the structural distance:

$$\kappa(T_a, T_b) = \kappa_S(T_a, T_b) / \sqrt{\kappa_S(T_a, T_a) \cdot \kappa_S(T_b, T_b)}$$

**Lemma 4.3:** The structural distance is a metric.

Proof: Subtree kernel $\kappa_S(T_a, T_b)$ is a valid kernel and we map the tree to one metric space. Consequently, we define a norm $\|T\| = \sqrt{\kappa(T, T)}$ based on the normalized kernel, and then the $L_2$ distance metric on this space is

$$L_2(T_a, T_b) = \|T_a - T_b\| = \sqrt{\kappa(T_a, T_a) + \kappa(T_b, T_b) - 2\kappa(T_a, T_b)} = \sqrt{2(1 - \kappa(T_a, T_b))}$$

The above deduction relies on $\kappa(T_a, T_a) = \kappa(T_b, T_b) = 1$. Finally, we conclude the $s_{dist}(T_a, T_b)$ is a metric since $L_2(T_a, T_b) = \|s_{dist}(T_a, T_b)\|$.

It is worth noting that classical approach of computing subtree kernel using suffix tree will lead to quadratic time cost [38]. We advocate that this will introduce large computational cost in our problem scenario since we need to build a suffix tree for each alternative plan at runtime. To address this problem, we first serialize a tree bottom-up, and then use a hash table to record the number of different subtrees. When calculating the subtree kernel, we can directly employ a hash table to find the same subtree. Although the worst-case time complexity is still quadratic, as we shall see in Section VII, it is significantly faster than the classical suffix tree-based approach in practice.

**Content Difference.** In order to further compare the content of the nodes, we adopt the edit distance to quantify content differences. Specifically, we first use the preorder traversal to convert a physical operator tree to a string, and based on it we apply the edit distance. Note that the alphabet is the collection of all node contents and the content of each tree node is regarded as the basic unit of comparison. For instance, in Fig. 3, the strings of $T_1$ and $T_2$ are ["Filter", "Table Scan[Table name]"] and ["Index Scan[Table name]"], respectively. The edit distance is 2 as they require at least two inserts/replaces to become identical to each other.

Since we want to ensure that the content distance is a metric, we further normalize the edit distance [26]. Let $\Sigma$ be the alphabet, $\lambda \notin \Sigma$ is the null string and $\gamma$ is a non-negative real number which represents the weight of a transformation. Given two strings $X, Y$, the normalized edit distance is

$$\text{edit}_\text{norm}(X, Y) = \frac{2 \cdot \text{edit}(X, Y)}{\alpha \cdot (|X| + |Y|) + \text{edit}(X, Y)}$$

where $\alpha = \max\{\gamma(a \rightarrow \lambda), \gamma(\lambda \rightarrow b), a, b \in \Sigma\}$ and $|X|$ is the length of $X$. This is proven to be a metric [26]. In our work, $\gamma = 1$. So the content distance can be defined as follows.

**Definition 4.4:** Given two query plans $\pi_a, \pi_b$ with corresponding tree structures $T_a, T_b$, based on the normalized kernel, we can define the **structural distance** between the plans as

$$s_{\text{dist}}(\pi_a, \pi_b) = \sqrt{1 - \kappa(T_a, T_b)}$$
We convert the differences in these types into binary form (Fig. 4(a)) where 1 indicates large. We can also represent the informative value of these types (based on learner feedback) in binary form (1 means informative). We can then employ a multivariate function fitting to find an appropriate relevance function such that alternative plans that are close to the informative ones receive high scores. Since 8 existing samples are insufficient for fitting a polynomial function with an order higher than 3 [21], it should appear in the following form:
\[
\rho_1 \times s + \rho_2 \times c + \rho_3 \times \text{cost} + \rho_4 \times s \times c + \rho_5 \times c \times \text{cost} + \rho_6 \times c \times \text{cost} + \rho_7 \times s \times c \times \text{cost}
\]
where \(\rho_1, \ldots, \rho_7\) are parameters to be fit. By fitting the above polynomial, we can define the relevance measure as follows.

**Definition 4.7:** Given the normalized structure distance \(s_i\), content distance \(c_i\) and cost distance \(\text{cost}_i\) between an alternative plan \(\pi_i\) and the QEP \(\pi^*\), the relevance score of \(\pi_i\) is defined as
\[
\text{rel}(\pi_i) = s_i + c_i + \text{cost}_i - s_i \cdot c_i - 2 \cdot s_i \cdot \text{cost}_i + 2 \cdot s_i \cdot c_i \cdot \text{cost}_i.
\]

The effect of \(\text{rel}(\cdot)\) is shown in Fig. 4(b). The red vertices represent the plan types that learners consider the most informative, which can effectively be captured by \(\text{rel}(\cdot)\). In addition, plans close to the informative ones receive high scores accordingly. In fact, each plan \(\pi\) can be referred to as a 3-dimensional vector in the space shown in Fig. 4(b), hence \(\text{rel}(\cdot)\) transforms a vector to a scale value, we shall use \(\text{rel}(\pi)\) instead of \(\text{rel}(\vec{\pi})\) for short.

**Remark.** Note that the distance and relevance scores are different. The former evaluates the differences between plans, while the latter is used to assign a higher score to AOPs that users consider informative. Any lower-order polynomial cannot satisfy this requirement. More importantly, these measures are generic and orthogonal to the distribution of plan categories and the characterization approach of informative alternative plans in specific applications.

## D. Plan Interestingness

Intuitively, through the plan interestingness measure, we aim to maximize the minimum relevance and distance of the selected query plan set. To this end, we can exploit measures that have been used in diversifying search results such as MaxMin, MaxSum, DisC, etc. [9; 10]. An experimental study in [11] has reported the effect of these choices in diversifying search results. Specifically, MaxSum and k-medoids fail to cover all areas of the result space; MaxSum tends to focus on the outskirts of the results whereas k-medoids clustering reports only central points, ignoring sparser areas. MaxMin and DisC perform better in this aspect. However, in the DisC model, users need to select a radius \(r\) instead of the number \(k\) of diverse results [10]. As the distribution of plans is uncertain, it is difficult to choose a suitable radius for the DisC model. Hence, we utilize the MaxMin model.
Suppose the plan set is \( \Pi \). Then the objective is to maximize the following condition where \( \lambda > 0 \) is a parameter specifying the trade-off between relevance and distance.

\[
(1 - \lambda) \min_{\pi_i \in \Pi} \text{rel}(\pi_i) + \lambda \min_{\pi_i, \pi_j \in \Pi} \text{dist}(\pi_i, \pi_j)
\] (1)

**Definition 4.8:** Given two query plans \( \pi_i, \pi_j \), the refined distance function is

\[
\text{Dist}(\pi_i, \pi_j) = \frac{(1 - \lambda)}{2} (\text{rel}(\pi_i) + \text{rel}(\pi_j)) + \lambda \text{dist}(\pi_i, \pi_j).
\]

**Lemma 4.9:** The distance function defined in Definition 4.8 satisfies the triangle inequality.

**Proof:** \( \text{dist}(\cdot) \) is a linear combination of \( s_{\text{dist}}, s_{\text{rel}} \) and \( \text{cost}_{\text{dist}} \). Because all of them is metric, \( \text{dist}(\cdot) \) is a metric too. \( \text{dist}(\cdot) \) satisfies the triangle inequality. Then we only need to prove that \( \frac{1 - \lambda}{2} (\text{rel}(a) + \text{rel}(b)) \) satisfies the triangle inequality. Let \( R(a,b) = \text{rel}(a) + \text{rel}(b) \), then

\[
R(a,b) + R(a,c) = \text{rel}(a) + \text{rel}(b) + \text{rel}(a) + \text{rel}(c) = 2 \cdot \text{rel}(a) + \text{rel}(b) + \text{rel}(c) = 2 \cdot \text{rel}(a) + R(b,c)
\]

Since \( \text{rel}(a) \geq 0 \), \( R(a,b) + R(a,c) \geq R(b,c) \). Similarly, we can prove that \( R(a,b) - R(a,c) \leq R(b,c) \). Therefore, the \( \text{Dist}(\cdot) \) consists of \( R(\cdot) \) and \( \text{dist}(\cdot) \) also satisfies the triangle inequality.

According to [14], the above bi-criteria objective problem can be transformed to find the refined distance defined in Definition 4.8. Consequently, we can define the plan interestingness \( U(\cdot) \) as follows.

**Definition 4.10:** Given a query plan set \( \Pi \), the plan interestingness is defined as \( U(\Pi) = \min \text{Dist}(\pi_i, \pi_j) \) where \( \pi_i \in \Pi, \pi_j \in \Pi \).

Now we consider \( U \) in Defn. 3.2, notably, \( U \)-TIPS allows a user to explore \( \Pi \) iteratively based on what has been revealed so far. We take into account the user preference in the form of \( r \in [1,0] \) (i.e., prefer or not) during the exploration. For example, a user may want to find only those plans with different join orders but not others. This scenario can be easily addressed by \( U \)-TIPS, which allows one to mark whether a newly generated plan is preferred. Consider Fig. 4b where each \( AP \) is represented as a 3-dimensional vector \( \vec{\pi} \) w.r.t. the QEP (\( \vec{\pi}^* = (0,0,0) \)). According to Defn. 4.7, a preferred \( AP \) (as well as its neighbors) should receive higher relevance scores (warmer in the heatmap). Intuitively, finding the category of the preferred \( AP \) (i.e., \( AP^*_1 \sim AP^*_V, AP^*_{V1} \)), say \( AP^*_1 \) (according to Fig. 4a, we can denote \( AP^*_1 = (0,0,1) \)), and moving the next plan towards \( (\text{resp.}, r = 1) \) if \( r = 1 \) (resp., \( r = 0 \), e.g., \( \vec{\pi} = \vec{\pi} + e(AP^*_1 - \vec{\pi}) \) would address this \( e = 0.05 \) by default). However, this strategy can affect only a single \( AP \) in the current iteration but not later. Therefore, instead of moving a single \( AP \), we select to update the origin of the 3-dimensional system, i.e., \( \vec{\pi}^* = \vec{\pi}^* - e(AP^*_1 - \vec{\pi}) \), where \( \vec{\pi} \) refers to the latest plan seen so far, and \( AP^*_1 \) refers to the category to which \( \vec{\pi} \) belongs. To find \( AP^*_1 \) for a plan \( \vec{\pi} \), we only need to perform a nearest neighbor search over the set of \( \{AP^*_1, AP^*_V, AP^*_{V1} \} \) (each can be encoded according to Fig. 4a) with respect to \( \vec{\pi} \).

**V. THE ARENA FRAMEWORK**

In this section, we present ARENA, an end-to-end framework for exploring informative alternative plans. The architecture of ARENA is shown in Fig. 5. It consists of the following components.

**ARENA GUI.** A browser-based visual interface that enables a user to retrieve and view information related to the QEP and alternative query plans of her input query in a user-friendly manner. Once an SQL query is submitted by a user, the corresponding QEP is retrieved from the RDBMS, based on which she may either invoke the B-TIPS or the I-TIPS component (detailed in Section VI).

**Candidate Plan Set Retriever.** In order to obtain the plan space for a given SQL query (i.e., \( \Pi^* \)), we investigate a series of existing tools including [15; 16; 27; 29; 39; 41] and finally we decided to adopt ORCA [39]. ORCA is a modular top-down query optimizer based on the cascades optimization framework [15]. We adopt it in ARENA for the following reasons. First, it is a stand-alone optimizer and is independent of the RDBMS a learner interacts with. Consequently, it facilitates portability of ARENA across different RDBMSs. Second, it interacts with the RDBMS through a standard XML interface such that the effort to support a new RDBMS is minimized. The only task one needs to undertake is to rewrite the parser that transforms a new format of query plan (e.g., XML in SQL Server, JSON in PostgreSQL) into the standard XML interface of ORCA. Consequently, this makes ARENA compatible with majority of RDBMSs as almost all of them output a QEP in a particular format such that a parser can be easily written without touching the internal of the target RDBMS. It is worth noting that the choice of implementation is orthogonal here. Other frameworks with similar functionality can also be adopted.

In order to implement the ORCA interface, we write all the information required by ORCA into an XML file, and then call the function in ORCA to read it for optimization. The XML file mainly consists of two parts: (a) the SQL query and (b) statistical information related to the query. For further information, please refer to [39] and its source code (https://github.com/greenplum-db/gporca). As a proof of concept, we
Algorithm 1: Algorithm IPS

Input: QEP $\pi^*$, collection of all AQP$s$ $\Pi^*$
Output: sampling of $\Pi^*$

1. $\Pi_1 \leftarrow \emptyset$
2. for $\pi_i \in \Pi^*$ do
   3. if $s_{dist}(\pi_i, \pi^*) = 0$ then
      4. $\Pi_1 = \Pi_1 \cup \{\pi_i\}$
   5. end
7. $\Pi_2 = UniformSampling(\Pi^*)$;
8. return $\Pi_1 \cup \Pi_2$;

have implemented this interface on top of PostgreSQL and Greenplum.

TIPS. The candidate plans from the above module are passed to the TIPS module. It implements the solutions proposed in the next section to retrieve the set of $k$ alternative query plans.

VI. TIPS ALGORITHMS

Recall from Section 1, we cannot integrate the selection of informative plans into the enumeration step of the query optimizer as these plans are selected based on the QEP and AQPs a learner have seen. Additionally, query optimizers search and filter on the basis of cost. In contrast, ARENA is neither seeking a single plan nor plans with the smallest cost. Hence, TIPS requires novel algorithms. Specifically, we present 1-TIPS and 2-TIPS to address the two variants of the problem introduced in Section III. We begin by introducing two pruning strategies that are exploited by them. Note that all these algorithms take as input a collection of candidate AQPs $\Pi^*$ that is retrieved by ARENA from the underlying RDBMS as detailed in Section V.

A. Group Forest-based Pruning (GFP)

Give that the plan space can be very large for some queries, traversing all AQPs to select informative plans can adversely impact efficiency of our solutions. Therefore, we need a strategy to quickly filter out uninformative AQPs. According to feedbacks from volunteers (Section II), $AP_{V1}$ and $AP_{V11}$ in Table I are uninformative. Observe that both these categories of plans have significant structural differences w.r.t. the QEP. Hence, a simple approach to filter uninformative plans is to calculate the difference between each AQ and the QEP and then discard AQPs with very large differences. Although this approach can reduce the number of plans, we still need to traverse the entire plan space for computing the differences. Hence, we propose a group forest-based pruning strategy, which can not only reduce the number of AQPs but also avoid accessing the entire plan space during the pruning process.

Cascade optimizers [15] utilize a MEMO structure (i.e., a forest of operator trees) to track all possible plans. A MEMO consists of a series of groups, each of which contains a number of expressions. An expression is employed to represent a logical/physical operator. Fig. 6a depicts an example of a MEMO, where each row represents a group and the square represents an expression in it. Each expression is associated with an id in the form of ‘GroupID.ExpressionID’, shown at the lower left corner (e.g., 3.1, 2.1). The lower right corner of an expression shows the id(s) of the child group(s) it points to.

Different expressions in the same group are different operators that can achieve the same functionality. For example, both 3.1 (HashJoin) and 3.2 (NTLoop) implement the join operation. Suppose Group 3 is the root group. When we traverse down from the expression in the root group, we can get a plan. For example the id sequence (3.1, 2.1, 1.1) represents the plan HashJoin(TableScan, TableScan). The upper right corner of each expression records the number of alternatives with this expression as the root node. For example, the number of expression 1.3 is 2, which includes 1.3→1.1 and 1.3→1.2.

Observe that the number of alternatives of the expression 3.1 in the root node is 8. That is, there are 8 different plans rooting at expression 3.1. However, there are only 2 tree structures over all these 8 plans as shown in Fig. 6b. The number in each node represents the id of the group. Since these trees represent connections between different groups, we refer to them collectively as group forest. Each group tree in a group forest represents multiple plans with the same structure and if the structure distance of a group tree and the QEP is larger than a given distance threshold $\tau_d$, i.e., $s_{dist}(\pi, \pi^*) \geq \tau_d$, all the corresponding ids can be filtered out. Observe that the number of group trees is significantly smaller than the number of AQPs. Therefore, uninformative plans can be quickly eliminated by leveraging them. In order to construct the group forest, we only need to traverse the MEMO in a bottom-up manner, which is extremely fast, i.e., time cost is linear to the height of the constructed tree.

Lastly, observe that if we filter based on $s_{dist}$ only, informative plans of types $AP_{IV}$ and $AP_{V11}$ (Table I) may be abandoned. The difference between these two types of plans and $AP_{IV}$ and $AP_{V11}$ is that the cost is smaller. Hence, we filter using both the structure and cost differences, i.e., $s_{dist}(\pi, \pi^*) \geq \tau_d, cost_{dist}(\pi, \pi^*) \geq \tau_c$ where $\tau_c$ is a predefined cost threshold.

B. Importance-based Plan Sampling (IPS)

It is prohibitively expensive to enumerate the entire plan space to select informative plans when the number of joins is large (i.e., involving more than 10 relations). We may sample some plans from the entire plan space and select informative plans from them. However, simply deploying random sampling may discard many informative plans. To address that, we propose an importance-based plan sampling (IPS) strategy that exploits the insights gained from feedbacks from volunteers in Section II to select plans for subsequent processing. The overall goal is to preserve the informative
plans as much as possible during the sampling. However, retaining all informative plans will inevitably demand traversal of every AQP to compute plan interestingness. Hence, instead of preserving all informative plans, we select to keep the most important ones when the number of joined relations in a query is large. We exploit the tradeoffs in Section II to sample plans. Observe that $AP_{II}$ plans receive a significant number of votes among all informative plans. Hence, we inject such plans as much as possible in our sample. Also, $AP_{II}$ plans have small structural differences with the QEP. Hence, given a QEP, we can easily determine whether an AQP belongs to $AP_{II}$ or not by finding all alternative plans with the same structure as QEP. These plans are then combined with uniformly random sampled results. The pseudocode of the IPS strategy is shown in Algorithm 1.

C. 1-TIPS Algorithm

Algorithm 2 outlines the procedure to address the 1-TIPS problem defined in Definition 3.2. If the number of joined relations is greater than the predefined IPS threshold $\tau_{i}$, we invoke IPS (Line 2). In addition, if the number of AQPs is greater than the predefined GFP threshold $\tau_{g}$, it invokes the group forest-based pruning strategy (Line 5), to filter out some uninformative plans early.

Notably, 1-TIPS allows a user to explore AP iteratively based on what has been shown. We take into account user-preference during the exploration. For instance, a user may want to find only those plans with different join orders but not others. Such scenario can be easily addressed by 1-TIPS, which enables one to mark whether a newly generated plan is preferred. Consider Fig. 4b where each AP is represented as a 3-dimensional vector $\vec{\pi}$ w.r.t. the QEP ($\vec{\pi}^{*} = (0, 0, 0)$). According to Definition 4.7, a preferred AP (as well as its neighbors) should receive higher relevance scores (warmer in the heatmap). Intuitively, finding the category (Line 9) of the preferred AP (i.e., $AP_{II} \sim AP_{IIV}, AP_{IIV}$), say $AP_{II}$ ($A^{P}_{II} = (0, 0, 1)$), and moving it towards it, e.g., $\vec{\pi} = \vec{\pi}^{*} + \epsilon (A^{P}_{II} - \vec{\pi})$) would address this ($\epsilon = 0.05$ by default). However, this strategy can only affect a single AP in the current iteration but not the ones afterwards. Therefore, instead of moving a single AP, we select to update the origin for the 3-dimensional system, i.e., $\vec{\pi}^{*} = \vec{\pi}^{*} - \epsilon (A^{P}_{II} - \vec{\pi})$ (Line 10), where $AP_{pre}$ refers to the category that $\pi$ belongs to.

D. B-TIPS Algorithm

We propose two flavors of the algorithm to address B-TIPS, namely, B-TIPS-BASIC and B-TIPS-HEAP.

B-TIPS-BASIC algorithm. We can exploit Algorithm 2 to address the B-TIPS problem. We first add the QEP $\pi^{*}$ to the final set $\Pi$. Then iteratively invoke the Algorithm 2 until there are $k+1$ plans in $\Pi$. Each time we just need to add the result of Algorithm 2 to $\Pi$. This algorithm has a time complexity of $O(k|\Pi^{*}|)$.

B-TIPS-HEAP algorithm. Since the minimum distance between plans in $\Pi^{*} - \Pi$ and $\Pi$ decrease monotonically with the increase of $|\Pi|$, we can use a max-heap to filter plans with very small distances to avoid unnecessary computation. Algorithm 3 describes the procedure. We first invoke our two pruning strategies (Line 1). In Line 6, we add a tuple which consists of the ID of an alternative plan $\pi$ and a distance to the heap, which indicates the minimum distance between plans in $\Pi^{*}$ and $\Pi$. Each time we choose a plan to add into $\Pi$, we do not update the distance of all tuples. Instead, we only need to check if this distance is correct (Line 16) and update it (Line 18). We select the alternative plan with the maximum minimum distance during each iteration. Observe that the worst-case time complexity is also $O(k|\Pi^{*}|)$. However, it is significantly more efficient than B-TIPS-BASIC in practice (detailed in Section VII).

For a MaxMin problem, a polynomial-time approximation algorithm provides a performance guarantee of $\rho \geq 1$, if for each instance of the problem, the minimum distance of the
There is a plan \( \pi \) such that \( \pi \in \Pi_{\text{OPT}} \). However, this contradicts our initial assumption that \( \pi \notin \Pi_{\text{OPT}} \). Therefore, the condition clearly holds after the \((k+1)^{th}\) addition.

Let \( \Pi' = \Pi_{\text{OPT}} \cap (\Pi^* \setminus \Pi_k) \). Note that \( k < |\Pi_{\text{OPT}}| \) since an additional plan is to be added. Thus \( \Pi' \geq 1 \). It is easy to verify that if \( |\Pi'| = 1 \), then the condition will hold after the \((k+1)^{th}\) addition. Therefore, assume that \( |\Pi'| = r \geq 2 \). Furthermore, let \( \Pi' = y_1, y_2, \ldots, y_r \). Since \( \Pi' \subseteq \Pi_{\text{OPT}} \), we must have \( \text{Dist}(y_i, y_j) \geq \text{OPT} \) for \( 1 \leq i, j \leq r, (i \neq j) \). Since the distances satisfy the triangle inequality (Lemma 4.9), it is possible to show that no two distinct plans in \( \Pi' \) can be blocked by the same node \( \pi_j \in \Pi_k \). Moreover, if there is a blocking plan in \( \Pi_k \) that also in \( \Pi_{\text{OPT}} \), the minimum distance of \( \Pi_{\text{OPT}} \) must be smaller than \( \text{OPT} / 2 \), because this is a blocking plan whose minimum distance with \( \Pi_{\text{OPT}} \) is smaller than \( \text{OPT} / 2 \). Therefore, none of the blocking plans in \( \Pi_k \) can be in \( \Pi_{\text{OPT}} \) and \( |\Pi'| = |\Pi_{\text{OPT}}| \). It is possible to show that \( \Pi_k \) must contain \( |\Pi_{\text{OPT}}| \) plans to block all plans in \( \Pi' \). However, this contradicts our initial assumption that \( k < |\Pi_{\text{OPT}}| \). Thus, the condition must hold after the \((k+1)^{th}\) addition.

**Remark.** Our proposed algorithms are generic and not limited to database education. First, they are orthogonal to any candidate plan set retrieval strategy. Second, the strategies GFP and IPS are designed based on informative and uninformative plan categories. Although in this paper we focus on categories relevant to database education application, it is easy to see that they can be easily adapted to other applications (e.g., database administration) where potentially different informative and uninformative plan categories may germinate from a human-centered characterization of AQP's. It is also easy to be adapted to a non-human-centered characterization of AQP's by changing the if-statement in Line 3 of Algorithm 1.

**VII. Performance Study**

In this section, we investigate the performance of ARENA. In the next section, we shall report the usefulness and application of ARENA in database education.

**A. Experimental Setup**

**Datasets.** We use two datasets. The first is the Internet Movie Data Base (IMDb) dataset [1; 24], containing two largest tables, cast_info and movie_info have 36M and 15M rows, respectively. The second one is the TPC-H dataset [2]. We use the TPC-H v3 and 1 GB data generated using dbgen [2].

Since the number of candidate plans impacts the performance of TIPS, which is mainly affected by the SQL statement, we choose different SQL queries to vary the plan size. For IMDb, [24] provides 113 SQL queries. We choose 8 of them for our experiments. For the TPC-H dataset, there are 22 standard SQL statements. Because the uneven distribution of the plan space of these queries, it is difficult to find SQL statements that are evenly distributed as in IMDb dataset. We select 6 SQL queries from them for our experiments. These selected queries and the plan space size are reported in Table II. Note that the space sizes are the entire plan spaces from ORCA.

**Baselines.** We are unaware of any existing informative plan selection technique. Hence, we are confined to compare ARENA with the following baselines. (a) We implemented a random selection algorithm (denoted by RANDOM). It is executed repeatedly \( n \) times and each time a result is randomly generated. The best result is returned after \( n \) iterations. In our experiment, we set \( n = 30 \). (b) A simple solution for TIPS is to return those with the least cost besides the QEP. Hence, we implement a cost-based approach that returns the top-\( k \) plans with the least cost in \( \Pi^* \) (denoted by COST) besides the QEP. (c) We also implemented an embedding-based approach (denoted by EMBEDDING), where we embed each AQP into a vector by feeding its estimated time cost and tree structure into an autoencoder following [4] and then rank the AQP's according to their Euclidean distance w.r.t. the QEP. Afterwards, the plan informativeness of the top-\( k \) can be computed following Defn. 4.10 accordingly.

All algorithms are implemented in C++ on a 3.2GHz 4-core machine with 16GB RAM. We denote B-TIPS-BASIC and B-TIPS-HEAP as B-TIPS-B and B-TIPS-H, respectively. Unless

---

**Table II: Dataset.**

| SQL     | Space Size | SQL     | Space Size |
|---------|------------|---------|------------|
| imdb_bb.sql | 2059       | imdb_bb.sql | 10200       |
| imdb_2a.sql  | 3211       | tpcch_18.sql | 2070        |
| imdb_5c.sql  | 3957       | tpcch_17.sql | 2864        |
| imdb_7b.sql  | 4975       | tpcch_11.sql | 11880       |
| imdb_7c.sql  | 7065       | tpcch_15.sql | 13800       |
| imdb_8b.sql  | 7887       | tpcch_21.sql | 24300       |
| imdb_8c.sql  | 9007       | tpcch_22.sql | 51870       |

---

1 We have put our system online at https://dbedu.xidian.edu.cn/arena/.
specified otherwise, we set $\alpha = 0.33$, $\beta = 0.33$, $\lambda = 0.5$, $\tau_l = 10$, and $\tau_d = 50000$.

**B. Experimental Results**

**Exp 1: Efficiency and effectiveness.** We first report the efficiency and effectiveness of different algorithms. For each query in Table II, we use different algorithms to select $k$ plans and report the plan informativeness $U$ and runtime, shown in Fig. 7(a-e) and 7(f-j), respectively. We make the following observations. First, the plan informativeness obtained by RANDOM, COST and EMBEDDING are significantly inferior to that of the B-TIPS algorithms. Moreover, the effectiveness of EMBEDDING decreases with respect to $k$ unless the volume of the plan space is large enough, i.e., $|\mathbb{P}| > 10000$. Second, the running time of B-TIPS-H is closer to COST and RANDOM, significantly faster than B-TIPS-B and EMBEDDING, and stable with increasing $k$ due to the heap-based pruning strategy.

**Exp 2: Comparison of strategies for subtree kernel.** As discussed in Section IV-B, there are two methods to compute the subtree kernel. We compare the efficiency of these two methods in initializing and computing the subtree kernel. The results are reported in Fig. 8. Observe that our hash table-based strategy is consistently more efficient.

**Exp 3: Impact of relevance.** In this set of experiments, we report the importance of the relevance measure (Defn. 4.7). We set $\lambda$ to 0, 0.5, and 1, and use B-TIPS to select 5 AQPs. The results are shown in Fig. 9. When $\lambda = 0$ (resp., $\lambda = 1$), $U$ is only composed of relevance (resp., distance). Hence, in Fig. 9(a), the selected plans differ much from QEP but have a small relevance. In Fig. 9(c), the selected plans have a high relevance value but exhibit a smaller difference. Based on the learner-centric characterization of AQPs in Section II, both of these cases are not desirable. When $\lambda = 0.5$, $U$ is composed of distance and relevance. Furthermore, it can be seen from Fig. 9(b) that when relevance and distance are considered together, these values are relatively reasonable and there is no case where a certain value is particularly small. The selected plans are not similar, while displaying large relevance. Therefore, this result is more reasonable and informative.

**Exp 4: Impact of GFP strategy.** Now we report the performance of the GFP-based pruning strategy. Fig. 10(a)-(c) plot the results on tpch22.sql, which have a large plan space. We compare the pruning power, plan informativeness, and efficiency of two variants of GFP, one pruned based only on $s_{\text{dist}}(\cdot)$ and the other with both $s_{\text{dist}}(\cdot)$ and $\text{cost}_{\text{dist}}(\cdot)$ (referred to as GFP and Cost, respectively). We vary the distance threshold $\tau_d$ and set the cost threshold $\tau_c = \tau_d$. Since the distribution of the plans is not uniform, the curves do not change uniformly. Obviously, pruning based only on $s_{\text{dist}}(\cdot)$ filters out more plans, but the plan informativeness is worse. If we do not use GFP strategy, the plan informativeness and time cost are 0.395 and 5s, respectively. That is, with GFP and Cost (resp., GFP), by saving more than 40% (resp., 95%) time cost, we may only sacrifice less than 6% (resp., 14%) of the plan informativeness. This justifies the reason for using both $s_{\text{dist}}(\cdot)$ and $\text{cost}_{\text{dist}}(\cdot)$ in GFP and Cost.

**Exp 5: Impact of the IPS strategy.** We investigate the effect of the IPS strategy described in Section VI. We choose 13c.sql in [24] which has 11 joins. We construct two sets of data: (a) We randomly select $n(n = 1000, 2000, \ldots, 5000)$ plans from all the plans currently available as the baseline data. (b) We deploy the IPS strategy. We find all alternative plans that have the same structure as QEP, add them to the $n$ randomly selected alternative plans. Notably, for a fair comparison, we randomly delete the same number of plans from (b) to ensure that the two sets of data, i.e., (a) and (b), have the same size of plan space. After that, we execute B-TIPS on the two sets of data and compare the plan informativeness. Fig. 10(c) and 10(d) report the results. Observe that for most cases IPS samples plans with higher plan informativeness.
Fig. 9: Effect of relevance.

VIII. ARENA IN DATABASE EDUCATION

It is well-established in education that effective use of technology has a positive impact on learning [17]. Technology has a positive impact on learning [17]. Hence, learner-friendly tools are often desirable to augment the traditional modes of learning (e.g., textbook, lecture). Indeed, database systems courses in major institutions around the world supplément traditional style of learning with the usage of off-the-shelf RDBMS (e.g., PostgreSQL) to infuse knowledge about database techniques used in practice. Unfortunately, these RDBMS are not designed for pedagogical purposes. Although they enable hands-on learning opportunities to build database applications and pose a wide variety of SQL queries over it, very limited learner-friendly support is provided beyond it [5]. Specifically, they do not expose a representative set of AQPs considered by the underlying query optimizer during the selection of a QEP in a user-friendly manner to aid learning. Hence, ARENA can potentially address this gap by facilitating retrieval and exploration of informative AQPs associated with a given query. In this section, we undertake a user study and academic assessment to demonstrate the usefulness of ARENA in database education. The various parameters are set to default (best) values as mentioned in the preceding section. In practice, these are set by administrators of ARENA.

A. User Study

We conducted a user study among CS undergraduate/postgraduate students who are enrolled in the database systems course in Xidian University. 51 unpaid volunteers participated in the study. Note that these volunteers are different from those who participated in the survey in Section II. After consenting to have their feedback recorded and analyzed, we presented a brief tutorial of the ARENA GUI describing the tool and how to use it. Then they were allowed to explore it for an unstructured time. Through the GUI, the participants can specify the choice of problem (B-TIPS vs I-TIPS), configure parameters (e.g., α, β, λ), and switch between subtree kernel and tree edit distance.

US 1: Survey. We presented 30 predefined queries on IMDb to all volunteers. A list of returned alternative plans is shown to the participants for their feedback. For our study, we generate the alternative plans using ARENA, COST, and RANDOM. All these plans can be visualized using the ARENA GUI. A participant can click on any returned alternative plan and ARENA will immediately plot it and the QEP side-by-side, where the tree structure, physical operator of each node, as well as the estimated cost at each node are presented. Moreover, each different node representing different operator or same operator with different cost in an alternative plan is highlighted in red or green, respectively. Volunteers were allowed to take their own time to explore the alternative plan space. Finally, they were asked to fill up a survey form, which consists of a series of questions. Where applicable, every subject is asked to give a rating in the Likert scale of 1-5 with higher rating indicating greater agreement to the affirmative response to the question or the technique/setting. We also instructed them that there is no correlation between the survey outcome and their grades. In the following, we report the key results.

Q1: Does ARENA facilitate understanding of the QEP selection process? Fig. 11a plots the results of the corresponding responses. Observe that 44 participants agree that the alternative plans returned by ARENA facilitate their understanding.

Q2: How well does ARENA/RANDOM/COST help in understanding the alternative plan space? In order to mitigate bias, the results of ARENA, COST, and RANDOM are presented in random order and participants were not informed of the approach used to select alternative plans. Fig. 11b reports the results. In general, ARENA receives the best scores (i.e., 36 out of 51 give a score over 3).

Q3: Preference for TIPS or B-TIPS. Among all the participants, 30 of them prefer to use B-TIPS whereas the remaining 13 prefer to use I-TIPS. The results justify that both these modes should be part of ARENA as different learners may prefer different mode for exploring alternative query plans.

Q4: How many alternative plans is sufficient to understand the QEP selection? Fig. 11c shows the responses. Most participants think that 10 or less alternative plans are sufficient.
Further feedbacks from these participants show that additional alternative plans exhibit limited marginal increase in their understanding. This also justifies that our approximate solution exhibits a diminishing return phenomenon.

Q5: Importance of structural, content, and cost differences. Fig. 11d reports the responses. Setting $\alpha$ (resp., $\beta$) as 0 indicates that the structural (resp., content) difference is not taken into account in ARENA. Similarly, by setting $\alpha = \beta = 0.5$, ARENA ignores the cost difference between plans. We observe that $\alpha = \beta = 1/3$ receives the best response. 36 participants give a score of 4 and above, while each of the remaining settings has less than 10 responses with scores 4-5. This highlights the importance of structural, content, and cost differences.

Q6: Importance of distance and relevance measures. $\lambda$ allows us to balance the impact of distance and relevance in selecting alternative plans. Fig. 11e reports the results. Clearly, by setting $\lambda = 0.5$, the results of ARENA is significantly better than either disabling distance or relevance.

Q7: Tree Edit Distance vs Subtree Kernel. Fig. 11f reports the results. In general, subtree kernel-based structural difference computation strategy receives superior scores (i.e., 36 out of 51 give a score over 3) to tree edit distance-based strategies (i.e., 27 out 51). This justifies the choice of the former as the default structural difference computation technique in ARENA.

US 2: Impact of different distance components. We explore the impact of different distance components on alternative plan preferences. We consider various combinations of distance components ($x$ axis in Fig. 11g and 11h). For each combination, the distance components are of equal weight. These combinations were hidden from the learners to mitigate any bias. We choose 3 SQL queries and ask the participants to mark and order the top-5 alternative plans (from 50 plans returned by ARENA under each setting) that they most want to see for each query. We then compare the user-marked list of plans in each case with the results returned by ARENA. For B-TIPS, we determine whether the returned result set contains the alternative plans that users expect to see and the number of such plans. We do not consider the order in which these plans appear. Therefore, we use the Mean P@$k$ ($k=5$) [30] to measure similarity. I-TIPS, on the other hand, returns alternative plans iteratively. The order of appearance is an important factor here. Therefore, for I-TIPS, we use the MAP@$K$ ($k=5$) [30].

The experimental results are reported in Fig. 11g and 11h. Observe that regardless of the exploration mode, the effect when all the three distances (i.e., structural, content and cost) are considered at the same time is superior to other combinations. Interestingly, when the weight of cost distance is large, the values of Mean P@$k$ and MAP@$k$ are both 0, which means that the results that users expect to see cannot be obtained in this setting. That is, only considering the cost of plans as a factor for selecting alternative query plans will result in potentially sub-optimal collection as far as learners are concerned.

US 3: Result difference between I-TIPS and B-TIPS. Lastly, we investigate the differences in the returned results of the two proposed algorithms. We collected 80 groups of results from 20 volunteers, each of them is requested to use both I-TIPS and B-TIPS and view 10 AQP. Notably, in I-TIPS, for each returned AQP, users can mark whether it is useful to them and the AQP returned in the subsequent steps take this feedback into account. We observe that on average 56.07% of the returned AQP are different in both solutions.
B. Academic Outcomes

The preceding section demonstrates that volunteers find ARENA useful in understanding the QEP selection process. In this section, we report the outcome of a quiz that test their understanding of this process. 50 learners from Xidian University participated in the quiz. These volunteers are different from those in the preceding section. They were given queries or query plans and were asked to answer questions on the followings: (a) other alternative plans that can also answer the query; (b) whether a plan is better than another; and (c) what plans the optimizer will never adopt. The questions can be found in appendix. In order to understand whether the adoption of ARENA useful to learners, we randomly separate them into a pair of test groups, each involving 25 students. The first group (referred to as GP1) does not have any experience with ARENA, but they were allowed to replay lecture video and review textbook material; and the second one (GP2) were exposed to ARENA before attending the quiz. Both groups are given the same time before the quiz, which are conducted simultaneously. The answer scripts of both groups were randomly mixed and graded by a teaching assistant, who is not involved with this research, in a double anonymity manner.

The distributions of scores for two questions in the quiz relevant to the study are depicted in Fig. 12. Observe that for both questions the performance of GP2 is significantly better than GP1. The average scores of learners in GP1 and GP2 are 7.76 and 8.24 for Question 1 (4.12 and 6.04 for Question 2), respectively. We also run a t-test for the assumption that the average score of GP2 is better in both questions. The p-values for the questions are 0.032 and 0.000303, respectively. In summary, GP2 significantly outperforms GP1 in the quiz.

Remark. We investigate the login and query logs of users since the system is open for public academic usage. The number of usage among the students in our University within recent two semesters are consistently above 100 until the holiday in each month. Among that, the learners have posed queries including nested subqueries, clustered functions, and multi-table (more than three table) joins. These facts demonstrate that learners actively turn to our system to improve their understanding of the process of plan exploration, especially for complex queries.

IX. RELATED WORK

Our work is related to the diversified top-k query, which aims to compute top-k results that are most relevant to a user query by taking diversity into consideration. This problem has been extensively studied in a wide variety of spectrum, such as, diversified keyword search in databases [45], documents [3] and graphs [13], diversified top-k pattern matchings [12], cliques [44] and structures [19] in a graph, and so on [42; 46]. In addition, [9] provides a comprehensive survey of different query result diversification approaches, [8] analyzed the complexity of query result diversification, and [14] used the axiomatic approach to analyze diversification systems. However, these techniques cannot be directly adopted here as they do not focus on informativeness of query plans in an RDBMS.

Given the challenges faced by learners to learn SQL [33], there has been increasing research efforts to build tools and techniques to facilitate comprehension of complex SQL statements [7; 18; 22; 25; 31; 32]. There are also efforts to visualize SQL queries [23; 25; 32] and query plans (e.g., [36]). However, scant attention has been paid to explore technologies that can enable learning of relational query processing [5; 16; 28; 43]. NEURON [28] generates natural language descriptions of QEPs using a rule-based technique and a question answering system to seek answers w.r.t. a QEP. Using deep learning methods, LANTERN [43] enhances NEURON to generate diversified natural language description of a QEP. MOCHA [40] is a tool for learner-friendly interaction and visualization of the impact of alternative physical operator choices on a selected QEP for a given SQL query. In contrast to this work, MOCHA requires that the learner has clear preferences on the physical operators they wish to explore w.r.t. a QEP. Picasso [16] depicts various visual diagrams of different QEPs and their costs over the entire selectivity space. In summary, ARENA complements these efforts by facilitating exploration of informative AQPs.

X. CONCLUSIONS AND FUTURE WORK

This paper presents a novel framework called ARENA that judiciously selects informative alternative query plans for an SQL query in order to enhance one’s comprehension of the plan space considered by the underlying query optimizer during QEP selection. To this end, ARENA implements efficient approximate algorithms that maximize the plan interestingness of the selected plans w.r.t. the QEP or already-viewed plan set. A key use case of ARENA is for pedagogical support where it can facilitate comprehension of the query execution process in an off-the-shelf RDBMS. Our performance study and user study with real-world learners indeed demonstrate the effectiveness of ARENA in facilitating comprehension of the alternative plan space. As part of future work, we plan to undertake a longitudinal study on the impact of ARENA on learners’ academic performance on this topic. Also, we wish to explore techniques to facilitate natural language-based interaction with the plan space.

REFERENCES

[1] Internet Movie Data Base (IMDb). http://www.imdb.com/.
[2] TPC-H data set. http://www.tpc.org/tpch/.
[3] Albert Angel and Nick Koudas. 2011. Efficient diversity-aware search. In Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2011, Athens, Greece, June 12-16, 2011. ACM, 781–792.
[4] Sanjeev Arora and Yingyu Liang and Tengyu Ma. 2017. A Simple but Tough-to-Beat Baseline for Sentence Embeddings. In 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. OpenReview.net.
[5] S. S. Bhowmick, H. Li. Towards Technology-Enabled Learning of Relational Query Processing. *IEEE Data Engineering Bulletin*, 45(3), 2022.

[6] S. Chaudhuri. An Overview of Query Optimization in Relational Systems. In *PODS*, 1998.

[7] J. Danaparamita, W. Gatterbauer. QueryViz: Helping Users Understand SQL Queries and Their Patterns. In *EDBT*, 2011.

[8] Ting Deng and Wenfei Fan. 2013. On the Complexity of Query Result Diversification. *Proc. VLDB Endow.* 6, 8 (2013), 577–588.

[9] Marina Drosou and Evaggelia Pitoura. 2010. Search result diversification. *ACM SIGMOD Record* 39, 1 (2010), 41–47.

[10] Marina Drosou and Evaggelia Pitoura. 2012. DisC diversity: result diversification based on dissimilarity and coverage. *Proc. VLDB Endow.* 6, 1 (2012), 13–24.

[11] Marina Drosou and Evaggelia Pitoura. 2013. POIKILO: A Tool for Evaluating the Results of Diversification Models and Algorithms. *Proc. VLDB Endow.* 6, 12 (Aug. 2013), 1246–1249.

[12] Wenfei Fan, Xin Wang, and Yinghui Wu. 2013. Diversified Top-k Graph Pattern Matching. *Proc. VLDB Endow.* 6, 13 (2013), 1510–1521.

[13] Konstantin Golenberg, Benny Kimelfeld, and Yehoshua Sagiv. 2008. Keyword proximity search in complex data graphs. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008. ACM, 927–940.

[14] Sreenivas Gollapudi and Anesh Sharma. 2009. An Axiomatic Approach for Result Diversification. In *Proceedings of the 18th International Conference on World Wide Web* (Madrid, Spain) (WWW ’09). Association for Computing Machinery, 381–390.

[15] Goetz Graefe. 1995. The Cascades Framework for Query Optimization. *IEEE Data Eng. Bull.* 18, 3 (1995), 19–29.

[16] Jayant R. Haritsa. 2010. The Picasso Database Query Optimizer Visualizer. *Proc. VLDB Endow.* 3, 2 (2010), 1517–1520.

[17] S. Higgins, Z. M. Xiao, M. Katsipatak. The Impact of Digital Technology on Learning: A Summary for the Education. *Education Endowment Foundation*, 2012.

[18] Y. Hu, Z. Miao, Z. Leong, H. Lim, Z. Zheng, S. Roy, K. Stephens-Martinez, J. Yang. I-Rex: An Interactive Relational Query Debugger for SQL. In *ACM Technical Symposium on Computer Science Education* (SIGCSE), 2022.

[19] Xin Huang, Hong Cheng, Rong-Hua Li, Lu Qin, and Jeffrey Xu Yu. 2013. Top-K Structural Diversity Search in Large Networks. *Proc. VLDB Endow.* 6, 13 (2013), 1618–1629.

[20] Toshihide Ibaraki and Tiko Kameda. 1984. On the optimal nesting order for computing r-relational joins. *ACM Transactions on Database Systems (TODS)* 9, 3 (1984), 482–502.

[21] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2013. *An introduction to statistical learning*. Springer.

[22] A. Kokkalis, P. Vagenas, A. Zervakis, A. Simitis, G. Koutrika, Y. E. Ioannidis. Logos: A System for Translating Queries into Narratives. In *SIGMOD*, 2012.

[23] W. Gatterbauer et al. Principles of Query Visualization. *IEEE Data Engineering Bulletin*, September 2022.

[24] Viktor Leis, Andrey Gubichev, Atanas Mirchev, Peter A. Boncz, Alfons Kemper, and Thomas Neumann. 2015. How Good Are Query Optimizers, Really? *Proc. VLDB Endow.* 9, 3 (2015), 204–215.

[25] A. Leventidis, J. Zhang, C. Dunne, W. Gatterbauer, H. V. Jagadish, and M. Riedewald. QueryVis: Logic-based Diagrams help Users Understand Complicated SQL Queries Faster. In *SIGMOD*, 2020.

[26] Yujian Li and Bi Liu. 2007. A Normalized Levenshtein Distance Metric. *IEEE Trans. Pattern Anal. Mach. Intell.* 29, 6 (2007), 1091–1095.

[27] Zhan Li, Olga Papaemmanouil, and Mitch Cherniack. 2016. OptMark: A Toolkit for Benchmarking Query Optimizers. In *Proceedings of the 25th ACM International Conference on Information and Knowledge Management*, CIKM 2016, Indianapolis, IN, USA, October 24-28, 2016. ACM, 2155–2160.

[28] Siyuan Liu, Sourav S Bhowmick, Wanlu Zhang, Shu Wang, Wanyi Huang, and Shafiq Joty. 2019. NEURON: Query Execution Plan Meets Natural Language Processing For Augmenting DB Education. In *SIGMOD ’19: International Conference on Management of Data*, Amsterdam, The Netherlands, June 30-July 5, 2019. ACM, 1953–1956.

[29] Ryan Marcus, Parimarjan Negi, Hongzi Mao, Nesime Tatbul, Mohammad Alizadeh, and Tim Kraska. 2021. Bao: Making Learned Query Optimization Practical. In *SIGMOD ’21: International Conference on Management of Data, Virtual Event, China*, June 20-25, 2021. ACM, 1275–1288.

[30] Brian McFee and Gert RG Lanckriet. 2010. Metric learning to rank. In *ICML*.

[31] Zhengjie Miao, Sudeepa Roy, Jun Yang. ExplainingWrong Queries Using Small Examples. *SIGMOD*, 2019.

[32] D. Miedema, G. Fletcher. SQLVis: Visual Query Representations for Supporting SQL Learners. In *VL/HCC*, 2021.

[33] D. Miedema, E. Aivaloglou, G. Fletcher. Identifying SQL Misconceptions of Novices: Findings from a Think-Aloud Study. In *ICER*, 2021.

[34] Mateusz Pawlik and Nikolaus Augsten. 2015. Efficient Computation of the Tree Edit Distance. *ACM Trans. Database Syst.* 40, 1, Article 3 (March 2015), 40 pages.

[35] S. S. Ravi, Daniel J. Rosenkrantz, and Giri Kumar Tayi. 2018. Approximation Algorithms for Facility Dispersion. In *Handbook of Approximation Algorithms and Metaheuristics, Second Edition, Volume 2: Contemporary and*
Emerging Applications. Chapman and Hall/CRC.

[36] D. Scheibli, C. Dinse, A. Boehm. QE3D: Interactive Visualization and Exploration of Complex, Distributed Query Plans. In Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, 877-881, ACM, 2015.
[37] John Shawe-Taylor and Nello Cristianini. 2004. *Kernel Methods for Pattern Analysis*. Cambridge University Press.
[38] Alex Smola and S.v.n. Vishwanathan. 2003. Fast Kernels for String and Tree Matching. In *Advances in Neural Information Processing Systems*, Vol. 15.
[39] Mohamed A. Soliman, Lyublena Antova, Venkatesh Raghavan, Amr El-Helw, Zhongxian Gu, Entong Shen, George C. Caragea, Carlos Garcia-Alvarado, Foyzur Rahman, Michalis Petropoulos, Florian Waas, Sivaramakrishnan Narayanan, Konstantinos Krikellas, and Rhonda Baldwin. 2014. Orca: a modular query optimizer architecture for big data. In *International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, June 22-27, 2014*. ACM, 337–348.
[40] J. Tan, D. Yeo, R. Neoh, H.-E. Chua, S. S. Bhowmick. MOCHA: A Tool for Visualizing Impact of Operator Choices in Query Execution Plans for Database Education. *PVLDB*, 15(12), 2022.
[41] Florian Waas and César A. Galindo-Legaria. 2000. Counting, Enumerating, and Sampling of Execution Plans in a Cost-Based Query Optimizer. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA*. ACM, 499–509.
[42] Bin Wang, Rui Zhu, Xiaochun Yang, and Guoren Wang. 2018. Top-K representative documents query over geotextual data stream. *World Wide Web* 21, 2 (2018), 537–555.
[43] Weiguo Wang, Sourav S. Bhowmick, Hui Li, Shafiq R. Joty, Siyuan Liu, and Peng Chen. 2021. Towards Enhancing Database Education: Natural Language Generation Meets Query Execution Plans. In *SIGMOD ’21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021*. ACM, 1933–1945.
[44] Long Yuan, Lu Qin, Xuemin Lin, Lijun Chang, and Wenjie Zhang. 2015. Diversified top-k clique search. In *31st IEEE International Conference on Data Engineering, ICDE 2015, Seoul, South Korea, April 13-17, 2015*. IEEE Computer Society, 387–398.
[45] Feng Zhao, Xiaolong Zhang, Anthony KH Tung, and Gang Chen. 2011. Broad: Diversified keyword search in databases. Technical Report.
[46] Rui Zhu, Bin Wang, Xiaochun Yang, Baihua Zheng, and Guoren Wang. 2017. SAP: Improving Continuous Top-K Queries Over Streaming Data. *IEEE Trans. Knowl. Data Eng.* 29, 6 (2017), 1310–1328.
A. Question 1
There is a SQL and the corresponding best plan (plan A) and an alternative plan (Plan B). Give some reasons why Plan A is better than Plan B. Explain each reason.

```sql
SELECT student.name
FROM student, takes, course
WHERE course.title = 'Image Processing'
  AND course.c_id = takes.c_id
  AND takes.year = 2002
  AND student.id = takes.id
  AND student.id < 1000;
```

**B. Question 2**
Consider the following SQL query that finds all courses offered by Professor Lembr in the spring.

```sql
SELECT title
FROM course, teaches, instructor
WHERE course.c_id = teaches.c_id
  AND teaches.id = instructor.id
  AND instructor.name = 'Lembr'
  AND teaches.semester = 'Spring';
```

**TABLE III: Statistics of relations.**

| Relation                  | Cardinality | Number of pages | Primary key          |
|---------------------------|-------------|-----------------|----------------------|
| course(c_id, title, credits) | 20000       | 50              | c_id                 |
| teaches (id, semester, c_id, year) | 20000000   | 34000           | (id, semester, c_id, year) |
| instructor (id, name)     | 5000        | 10              | id                   |

Assume that:
- teaches.id is a foreign key that references instructor.id.
- teaches.c_id is a foreign key that references course.c_id.
- teaches are evenly distributed on id and semester and semester can only be Spring or Fall.
- There is only one instructor named Lember.
- For each table, there is a B+ tree index on its primary key and all index pages are in memory.
- All intermediate tables can be stored in memory.

1) Question: : compute the worst cost of the physical query plan in Fig. 14.
2) Question: : Draw a query plan for the above query that the query optimizer will NOT consider. Justify your answer.