A MISSING PROOFS IN SECTIONS 4 AND 5

A.1 Proof of Lemma 2

Lemma 2 tells the value ranges of n elements given their average value x, i.e., the i-th element must fall in $[a_i, b_i]$ where $a_i = \max\{x_l, \frac{x \cdot n - \sum_{j=i+1}^{n} x_h}{i}\}$ and $b_i = \min\{x_h, \frac{x \cdot n - \sum_{j=1}^{i-1} x_l}{n-i+1}\}$. It can be proved as follows.

Proof of Lemma 2. Given a non-decreasing sequence $\mathbf{x} = (x_1, \cdots, x_n)$ with $x_i \in [x_l, x_h]$ and $avg(\mathbf{x}) = x$, assume that there exists a element x_j in $\mathbf{x}$ is out of $[a_j, b_j]$, i.e., $x_j \notin [a_j, b_j]$. There are four cases to prove that such the x_j cannot exist.

Case 1: $a_j = x_l$ and $x_j < a_j$. In this case, x_j is less than the lowest value of the domain, which violates the constraint $x_j \in [x_l, x_h]$.

Case 2: $b_j = x_h$ and $x_j > b_j$. Similarly, the constraint $x_j \in [x_l, x_h]$ is violated.

Case 3: $a_j > x_l$ and $x_j < a_j$. One can construct a new sequence $\hat{\mathbf{x}} = \{a_j, \cdots, a_j, x_h, x_h, \cdots, x_h\}$, where the first j elements are a_j and the rest of the elements are equal to x_h. As $\mathbf{x}$ is non-decreasing, we know that each element in $\hat{\mathbf{x}}$ is no less than that in $\mathbf{x}$. Furthermore, there is $avg(\hat{\mathbf{x}}) > avg(\mathbf{x})$ due to $x_j < a_j$. However, based on Eq. 4, there are $a_j = \frac{x \cdot n - \sum_{k=j+1}^{n} x_h}{j}$ and $avg(\hat{\mathbf{x}}) = x$. It contradicts the fact that $avg(\mathbf{x}) = x$.

Case 4: $b_j < x_h$ and $x_j > b_j$. Similarly, we can also construct a new sequence $\hat{\mathbf{x}} = \{x_l, \cdots, x_l, b_j, \cdots, b_j\}$, where the first $(j-1)$ elements are x_l and other elements equal b_j. For the sequence $\hat{\mathbf{x}}$, its average value is less than $avg(\mathbf{x})$, since each element in $\hat{\mathbf{x}}$ is no higher than that in $\mathbf{x}$ and there is $x_j > b_j$. Besides, we get $avg(\hat{\mathbf{x}}) = x$ based on Eq. 4, which is in conflict with $avg(\mathbf{x}) = x$.

Considering all cases together, the existence of $x_j \notin [a_j, b_j]$ is a contradiction. As a result, the value range of any element x_i in the non-decreasing sequence $\mathbf{x}$ is $[a_i, b_i]$. The proof is complete. □

A.2 Proof of Theorem 2

We prove Theorem 2 case by case.

Proposition 4. *Given the selection query Q_1:* `select` A `from` R_j `where predicates`*, the avg-count query Q_2 and the count query Q_3 with the same predicates, Q_1 determines Q_2 and Q_2 determines Q_3. Further, we have that $p_b(Q_3) \leq p_b(Q_2) \leq p_b(Q_1)$.*

Proof. Obviously, Q_1 determines Q_2 and Q_2 determines Q_3 due to the semantics of these queries. Moreover, for each tuple $t \in R_j$, there is $E_t(Q_1) \subseteq E_t(Q_2) \subseteq E_t(Q_3)$. We can prove this step by step.

First, when t satisfies the predicates of these three queries, its value ranges under the three queries Q_1, Q_2, Q_3 are $E_t(Q_1) = \{v | v \in S_j \text{ and } Q_1(v) = t.A\}$, $E_t(Q_1) = \{v | v \in S_j \text{ and } Q_1(v) \in [a_i, b_i]\}$, and $E_t(Q_1) = \{v | v \in S_j \text{ and } Q_1(v) \neq \varnothing\}$, respectively. $[a_i, b_i]$ is the value range derived by Lemma 2. It is easy to find that $E_t(Q_1) \subseteq E_t(Q_2) \subseteq E_t(Q_3)$ holds.

Second, the value ranges of t under the three queries are same as $E_t = \{v | v \in S_j \text{ and } Q_1(v) = \varnothing\}$. In this case, $E_t(Q_1) \subseteq E_t(Q_2) \subseteq E_t(Q_3)$ also holds. Considering all cases together, there is $E_t(Q_1) \subseteq E_t(Q_2) \subseteq E_t(Q_3)$ and the query price has $p_b(Q_3) \leq p_b(Q_2) \leq p_b(Q_1)$. It completes the proof. □

Proposition 5. *Suppose there are two queries with the same predicates, i.e., Q_1:* `select` A `from` R_j `where predicates` *and the avg-count query Q_2:* `select avg(`A`), count(*) from` R_j `where predicates`*. When the number of qualified tuples of Q_1 is $n \leq 1$, the two queries are equivalent and have the same price.*

Proof. First, when $n = 0$, the two queries have empty answers and they are equivalent. In this case, all tuples in R_j under Q_1 and Q_2 have the same value range $E_t = \{v | v \in S_j \text{ and } Q_1(v) = \varnothing\}$. Thus, the prices of Q_1 and Q_2 are the same.

Second, when $n = 1$, Q_1 tells that, the value range of the one qualified tuple t is $E_t(Q_1) = \{v | v \in S_j \text{ and } Q_1(v) = t.A\}$ and those of the rest of tuples are $E_t = \{v | v \in S_j \text{ and } Q_1(v) = \varnothing\}$. Under Q_2, the value range of the qualified tuple t on A can be computed based on Eq. 4, which is the same as $E_t(Q_1)$. Also, the rest of the tuples have the same value range E_t under Q_2. It means that, Q_1 and Q_2 have the same price. It completes the proof. □

Proposition 6. *Suppose there are two queries with the same predicates, i.e., Q_1:* `select` A `from` R_j `where predicates` *and the avg-count query Q_2:* `select avg(`A`), count(*) from` R_j `where predicates`*. When they both have the predicate '$A = c$' and c is a constant, the two queries are equivalent and have the same price.*

Proof. When Q_1 and Q_2 have the same predicates $A = c$ where c is a constant, the query answer of Q_2 is (c, n), while Q_1 returns a size-n result $(c, c, \cdots, c)$. In this case, the value ranges of n tuples satisfying the predicates are $E_t(Q_1) = \{v | v \in S_j \text{ and } Q_1(v) = c\}$. On the other hand, one can compute the value ranges of these tuples under Q_2 based on Eq. 4. In this case, the a_i and b_i derived from Eq. 4 are c, since the minimum, the maximum, and the average of these tuples are c. Thus, these n tuples have the same value range, i.e., $E_t(Q_1)$. Based on the same tuple information, Q_1 and Q_2 have the same price. The proof is complete. □

Proposition 7. *Suppose there are two queries with the same predicates, i.e., Q_1:* `select` A `from` R_j `where predicates` *and the avg-count query Q_2:* `select avg(`A`), count(*) from` R_j `where predicates`*. When all qualified tuples have the same value on the attribute A, the price of Q_2 equals that of Q_1.*

Proof. When Q_1 and Q_2 have the same predicates $A = c$ where c is a constant, the query answer of Q_2 is (c, n), while Q_1 returns a size-n result $(c, c, \cdots, c)$. In this case, the value ranges of n tuples satisfying the predicates are $E_t(Q_1) = \{v | v \in S_j \text{ and } Q_1(v) = c\}$. Specifically, the value ranges of the n qualified tuples under Q_2 are computed based on Eq. 4. As these tuples have the same value on the attribute A, the maximum and minimum of the n tuples, i.e., x_l and x_h, are the same as the average value. Hence, the a_i and b_i derived from Eq. 4 are also the same as the average, i.e., their real value. Thus, the value range of each qualified tuple under Q_1 reveals its real value on the attribute A, which is the same as that under Q_1. In light of this, all tuples under Q_1 and Q_2 have the same value range and thus they have the same price. The proof is complete. □

Lemma 4. *Suppose there are four queries with the same predicates, i.e., Q_1:* `select` A `from` R_j `where same predicates`*, the avg-count query Q_2:* `select avg(`A`), count(*) from` R_j `where same predicates`*, the max/min query Q_3:* `select min(`A`) from` R_j `where same predicates`*, and the count query Q_4:* `select count (*) from` R_j `where same predicates and` $A < \bar{x}$ *where $\bar{x}$ is* *the average value. When all qualified tuples have the same value on the attribute A, we have that Q_2 and Q_3 determine Q_1, while Q_2 and Q_4 determine Q_1 holds. Moreover, ARIA ensures no arbitrage, i.e., $p_b(Q_1) \leq p_b(Q_2) + p_b(Q_3)$ and $p_b(Q_1) \leq p_b(Q_2) + p_b(Q_4)$.*

Proof. Case 1. Q_2 tells the average of n elements and Q_3 tells their maximum/minimum. As these n elements have the same value, the maximum/minimum is the same as the average. In this case, data buyers can know based on Q_2 and Q_3 that, these elements must take the same value. Otherwise, the maximum/minimum cannot be the same as the average. In other words, Q_2 and Q_3 together tell the real values of these n elements, which are also revealed under Q_1. Hence, Q_2 and Q_3 determine Q_1 in this case.

Case 2. Q_2 tells the average of n elements and Q_4 tells the number of elements that are less than the the average. As these n elements have the same value, the answer of Q_4 is zero. Similarly, data buyers can know based on Q_2 and Q_4 that, these elements must take the same value. Otherwise, there must exist some elements less than the average. In this case, Q_2 and Q_3 actually tell the real values of these n elements, which are also revealed under Q_1. Thus, there is Q_2 and Q_3 determine Q_1.

Moreover, based on Proposition 7, there is $p_b(Q_1) = p_b(Q_2)$. Hence, there are $p_b(Q_1) \leq p_b(Q_2)+p_b(Q_3)$ and $p_b(Q_1) \leq p_b(Q_2)+ p_b(Q_4)$. The proof is complete. ◻

Before making effective proofs on the fourth case in Table 3, we first introduce the price of the limit query, i.e., the selection query with the `limit k` clause. The "`limit` k" means that only k selections are returned. When the result size is k, i.e., $Q(D) = \{v_1, \cdots, v_k\}$, the buyer can know the value domains of k tuples, i.e., $E_t(Q) = \{v \in S_j \text{ and } Q'(v) = v_i\}$ where Q' removes the limit clause from Q. For other $(N_j - k)$ tuples, they can either satisfy the query predicates or break the predicates, and thus their value domains are still S_j. When the result size is smaller than k, the limit query Q equals the query Q', since the real result size is known. In this case, the value domain of each tuple can be derived as $E_t(Q) = \{v | v \in S_j \text{ and } Q'(v) = Q'(t)\}$. Based on the features of the prices of the limit query and avg-count query, we prove Lemma 5 based on Proposition 8. It means that, ARIA ensures no arbitrage where the original values of n elements are determined by the average and the $(n-1)$ elements.

Proposition 8. *Given any avg-count query Q with answer $(\bar{x}, n)$ and $n \geq 2$, the total price of n qualified tuples under ARIA is no less than $|S_j| - 1$.*

Proof. There are at least two qualified tuples when $n \geq 2$. Let t_1 (resp. t_2) be the qualified tuple with minimum (resp. maximum) value on the aggregated attribute and $[x_l, x_h]$ be the domain of this attribute. Based on Theorem 2, the value domain of t_1 is $E_1 = \{v | v \in_j \text{ and } Q'(v) \in [x_l, x]\}$, while that of t_2 is $E_2 = \{v | v \in_j \text{ and } Q'(v) \in [x, x_h]\}$. Q' is the non-aggregate version of Q. The total information gain of these two tuples is $2 \cdot |S_j| - |E_1| - |E_2|$. As E_1 and E_2 only overlaps at v with $Q'(v) = x$, there is $|E_1| + |E_2| = |E| + 1$, where $E = \{v | v \in_j \text{ and } Q'(v) \in [x_l, x_h]\}$ and $|E| \leq |S_j|$, Therefore, the total information gain of all qualified tuples is no less than $|S_j| - 1$. The proof is complete. ◻

Lemma 5. *Given the queries Q_1: select A_k from R_j where predicates, Q_2: select A_k, count(*) from R_j where predicates and Q_3: select A_k from R_j where predicates limit $n-1$, i.e., Q_1 is determined by Q_2 and Q_3, there is $p_b(Q_1) \leq p_b(Q_2) + p_b(Q_3)$ where n is the number of qualified tuples in Q_1.*

Proof. When $n = 0$ or $n = 1$, the avg-count query Q_2 equals Q_1, and the derived tuple information of Q_2 in Section 4.2 captures such phenomenon. In these cases, the price of Q_1 is the same as that of Q_2 and there is no arbitrage. When n is higher than 1, the price of Q_1 considers the information gain of all tuples in R_j, i.e., n qualified tuples and $(N_j - n)$ unsatisfactory tuples. The information of unsatisfactory tuples is also revealed in Q_2, and their price under Q_1 is the same as that in Q_2. For the first $(n-1)$ qualified tuples, Q_1 and Q_3 both reveal their information, and these tuples are priced as same. Then, for the last qualified tuple t, its price in Q_1 is no higher than $|S_j| - 1$, since its value domain has at least one element, i.e., itself. Besides, Proposition 8 indicates that the total price of all qualified tuples in Q_2 is no less than $|S_j| - 1$. Overall, the price of Q_1 is no higher than the total price of Q_2 and Q_3 and there is no arbitrage caused by the $(n-1)$ freedom of the average value. The proof is complete. ◻

Based on the above propositions and lemma, we can prove Theorem 2 in the following.

Proof of Theorem 2. Given a non-decreasing sequence $\mathbf{x} = (x_1, \cdots, x_n)$ with $x_i \in [x_l, x_h]$ and $avg(\mathbf{x}) = x$, assume that there exists a element x_j in $\mathbf{x}$ is out of $[a_j, b_j]$, i.e., $x_j \notin [a_j, b_j]$. There are four cases to prove that such the x_j cannot exist.

Case 1: The selection query determines the avg-count query with the same predicates, while the avg-count query determines the count query. It is proved to ensure no arbitrage for such query determinacy by Proposition 4.

Case 2: The avg-count query can determine the selection query with the same predicates when the number of qualified tuples n is no higher than one or the predicates include the constant equation "$A = c$". The arbitrage-free property under this case is provided in Propositions 5 and 6.

Case 3: Proposition 7 ensures no arbitrage for the special case where all tuples satisfying the predicates of the avg-count query have the same value on the aggregated attribute.

Case 4. Lemma 4 achieves no arbitrage when the query determinacy comes from the degree of freedom of the average value.

As a result, ARIA is arbitrage-free to price avg-count queries considering all possible query determinacy in Table 3. The proof is complete. ◻

A.3 Proof of Lemma 3

Proof of Lemma 3. First, as Q is a selective query with the non-empty answer, there exists one tuple t_1 satisfying the query predicates and another tuple t_2 not. Then, for the tuple t_1, it can never take the value of t_2, i.e., $t_2 \notin E_{t_1}(Q)$ and $|S_j| - |E_{t_1}(Q)| > 0$. Hence, the price of t_1 under Q is positive and the query price is higher than zero. The proof is complete. ◻

Algorithm 5: The ARIA-SAJ Algorithm

Input: the query Q on relations R_Q; the sizes of all relations $N_1, \cdots, N_m$; the support sets $S_1, \cdots, S_m$
Output: the query price p
1: $Q' \longleftarrow$ remove the max/min/avg/sum/count keyword, remove the group by clause, and add the ID column of each involved relation $R_j \in R_Q$ into the selection clause of Q
2: obtain $Q'(D)$ and initialize the price p as zero
3: **foreach** $R_j \in R_Q$ **do**
4: $\quad Q_j(D) \leftarrow$ extract the results of $Q'(D)$ on relation R_j, remove repetitive results, and remove the ID column
5: $\quad Q_j(S_j) \leftarrow$ extract the results of $Q'(D - R_j + S_j)$ on relation S_j, remove repetitive results, and remove the ID column
6: $\quad$ **if** *the aggregated attribute in Q is in R_j* **then**
7: $\quad\quad$ **if** *min/max in Q* **then**
8: $\quad\quad\quad Q_j(D) \leftarrow$ group $Q_j(D)$ and compute the maximum/minimum of each group
9: $\quad\quad\quad p_j \longleftarrow$ ARIA-MM($Q_j(D), Q_j(S_j), N_j, |S_j|$)
10: $\quad\quad$ **else**
11: $\quad\quad\quad p_j \longleftarrow$ ARIA-AVG($Q_j(D), Q_j(S_j), N_j, |S_j|$)
12: $\quad$ **else**
13: $\quad\quad p_j \longleftarrow$ ARIA-Base($Q_j(D), Q_j(S_j), N_j, |S_j|$)
14: $\quad p \longleftarrow p + p_j$
15: **return** p

B MISSING ALGORITHM IN SECTION 5

We describe the missing ARIA-SAJ algorithm in Section 5 to price the select-aggregate-join (SAJ) queries. For simple count queries, the SAJ query can be rewritten as the SJ query, which replaces the `count(*)` with 1. For other types of aggregations, each SAJ query is decomposed into one select-aggregate (SA) query and multiple selection queries.

Algorithm 5 depicts the procedure of pricing SAJ queries in ARIA. It takes the SAJ query Q on multiple relations R_Q, the sizes of all relations (i.e., $N_1, \cdots, N_m$), and all support sets (i.e., $S_1, \cdots, S_m$) as inputs. It outputs the query price p. It first constructs the query Q' based on Q, which removes the special keywords and clauses (e.g., `max`, `min`, `avg`, `sum`, `count`, `group by`) and adds the ID column of each relation $R_j \in R_Q$ into the selection clause of Q (line 1). Next, it executes $Q'(D)$ to derive $Q_j(D)$ and initializes the query price p as zero (line 2). Then, ARIA-SPJ starts to derive the price of each single-relation query Q_j (lines 3-10). It first derives $Q_j(D)$ by manually extracting the results of $Q'(D)$ on R_j, removing the repetitive results due to the join operation, and dropping the ID column. Similarly, ARIA-SPJ can obtain the query answer $Q_j(S_j)$ (line 5). Q' is executed on the relation S_j and other relations (except for R_j) in D, while the query answer is denoted as $Q'(D - R_j + S_j)$.

Given $Q_j(D)$ and $Q_j(S_j)$, ARIA-SAJ starts to compute the price of each single-relation query. For the relation where the aggregate attribute is located, ARIA-SAJ needs to consider the price of the SA query (lines 7-11). In particular, ARIA-SAJ manually computes the max/min value of each group based on $Q_j(D)$ and employs ARIA-MM to compute the price of max/min query. When Q is the avg-count/sum-count query, ARIA-SAJ directly utilizes the ARIA-AVG algorithm in line 11. Otherwise, ARIA-SAJ employs ARIA-Base to compute the price p_j of the non-aggregate query Q_j (line 13). The derived price p_j is accumulated to the query price p. When all involved relations are considered, the query price p of the SAJ query Q is returned (line 15).

The complexity of pricing the max/min queries on joined tables is $\sum_{R_j \in R_Q} |S_j|$, since ARIA-Base and ARIA-MM both have linear time complexity. Moreover, it requires $O(\sum_{R_j \in R_Q} |S_j| \cdot \log \sum_{R_j \in R_Q} |S_j|)$ complexity of pricing avg-count/sum-count queries, the log-linear complexity $O(|S_j| \cdot \log |S_j|)$ of pricing avg-count queries.

Received 17 July 2024; revised 12 March 2024; accepted 5 June 2024