Similarity Grouping by Influence: Exploring Result Diversification in Similarity Group-by Operator

Willian D. Oliveira[1], **Anna J. C. Lauton**[2], **Caetano Traina Jr.**[1], **Lucio F. D. Santos**[2]

[1]Institute of Mathematics and Computer Sciences - University of São Paulo (USP)

[2]Federal Institute of Technology North of Minas Gerais (IFNMG)

{willian,caetano}@icmc.usp.br, lucio.santos@ifnmg.edu.br

Abstract. *The group-by operator groups the tuples sharing the same values in specified attributes, then extracts summaries from each group. However, several data stored by modern applications are best queried not by equality but by similarity, giving rise to a number of questions, such as: "How to obtain groups, such that each one contains the k tuples most similar?" or "How to include diversity in the results?". In this paper, we present a binary grouping operator focused on diversified similarity comparisons, which is able to answer such questions. We define the operator algebraically and show its applicability to enable the execution of grouping operations over complex attributes, such as multidimensional data. We provide an algorithm, called Similarity Grouping by Influence – SGIa — to implement the binary operator. An experimental evaluation performed on real data shows the SGIa is able to timely meet real application needs with significant results.*

1. Introduction

Grouping and aggregation are the main operations provided by relational database management systems (RDBMS) for decision support applications and data summarization [Silva et al. 2019]. The traditional Group-by command partitions a relation into several groups (*grouping*), in such a way that a subset of the relation attributes has the same ('=') values in every tuple of each group. Thereafter, each group is *aggregated* using any attributes.

Modern applications store data that are best queried not by equality but by similarity, that is, the attributes are compared regarding a measurement of the similarity of their features. As example, suppose a crowdsourcing app used by smart cities[1] where citizens report problems such as fly-tipping, broken paving slabs, and/or street lighting. The users can send information like photos, descriptions, and geo-coordinates. The images combined with the extra data may help the local government prioritize reports with high demands. Similarity grouping answers can focus on grouping by a range predicate over the geo-coordinates (returning elements that are at most ξ apart from the predicate), and summarizing the problem information such as maximal distance between the main report images and the newly reported ones.

Although several works have studied similarity search regarding different domains and operators, such as selection [Silva et al. 2019, Jasbick et al. 2020] and similar-

[1]FixMyStreet App: https://www.fixmystreet.com/

ity join [Santos et al. 2015, Yang et al. 2023], few of them have tackled similarity grouping and aggregation [Schallehn et al. 2004, Tang et al. 2016, Silva et al. 2019]. Moreover, none of them has been found useful when the data to be searched by similarity may have many elements too much similar to each other. For instance, in a smart cities application, as the same problem may be repeatedly registered by several citizens, many photos become almost identical copies of others already reported. In such cases, applying similarity grouping may find a problem to "seed" the groups, as too similar objects may split a large "seed" group into several ones, increasing the effort of analysis to identify distinct city problems.

A number of approaches have discussed the definition of result diversification on similarity operators [Lopes et al. 2021, Drosou et al. 2017]. For instance, Jasbick *et. al* (2020) explored k-nearest neighbors predicate on selections. Santos *et. al* (2015) proposed a diversified join operator to the similarity range predicate. A first approach about grouping is presented by Santos *et. al* (2016), but all of them focused on clustering operations, and do not discuss how to define group-by operator with aggregation. Apart from the search operator extended with diversity, the result diversification based on influence (RDI) [Lopes et al. 2021] has been explored to retrieve representatives elements that are similar to a query element and diverse among result elements, using a dynamic "separation distance" according to the data distribution around the elements, the so-called *influence* criteria [Lopes et al. 2021].

This paper introduces the Similarity Grouping by Influence (SGI) for RDBMSs, aiming at capturing a holistic and more diversified perspective of group "seeds". We present a formalization of similarity grouping that seamlessly integrates similarity with diversity consideration by means of a binary operator. Furthermore, we present an algorithm to ensure that the grouped elements are both similar to the seeds and dissimilar to other group representatives.

The remainder of this paper is organized as follows. Section 2 provides the background and related work, while Section 3 describes our proposal for diversified similarity grouping. Finally, Sections 4 and 5 provide the evaluation and conclude the study.

2. Background and Related Work

Similarity comparisons in RDBMS. A complex attribute S must be represented in a space defined as $M = \langle \mathbb{S}, d \rangle$, where $\mathbb{S}$ is the attribute domain ($Dom(\mathsf{S}) = \mathbb{S}$) and d is a distance function. The attribute's active domain $S = D^{*}o\, m\,(\mathsf{S})$ is the subset of values $t[\mathsf{S}] \in \mathbb{S}$ that occur in at least one tuple t of the relation. While scalar attributes are compared by θ operators based on identity and order relationship ($\theta \in \{=, \neq, <, \ldots\}$), complex attributes are essentially compared by two similarity relationship comparison operators: the similarity range and the k-nearest neighbor operators. Thus, given a complex attribute S such that $D^{*}o\, m\,(\mathsf{S}) = S$, $S \subseteq \mathbb{S}$, a query center $s_q \in \mathbb{S}$ and a predicate $\mathsf{S}\ \theta_s\ s_q$: (a) The range operator $\theta_s = Rng(d, \xi)$ returns `TRUE` for each element $s_i \in S$ iff $\delta s_i, s_q \leq \xi$; (b) The k-nearest neighbor operator $\theta_s = k\text{-}NN(d, k)$ returns `TRUE` for each element $s_i \in S$ iff s_i is one of the k elements closest to s_q, according to the distance d.

***Influence* measures.** Given two elements $s_i, s_j \in \mathbb{S}, s_i \neq s_j$, their mutual *Influence* (inverse dissimilarity) is calculated by $I(s_i, s_j) = 1/d(s_i, s_j)$. Given a query reference $s_q \in \mathbb{S}$, a diversified (*Influence*-free) neighbor $s_i \in S$, and a dataset object $s_j \in S$, then

their *Influence* measures define a ternary relationship that indicates s_j as more influenced by s_i than s_q *iff* $I(s_i, s_j) > I(s_j, s_q)$. Thus, s_j should not be considered for inclusion in result as s_i is already a diversified neighbor.

Influence Set. The *Influence Set* of a diversified neighbor s_i regarding a query reference $s_q \in \mathbb{S}$ encompasses every entry $s_j \in S \setminus \{s_i \cup s_q\}$ that are *(i)* farther from s_q than s_i and *(ii)* more *Influenced* by s_i than s_q, *i.e.*, $\ddot{I}_{s_i,s_q} = \{s_j \mid s_j \in S \setminus \{s_i, s_q\}, I(s_i, s_j) > I(s_i, s_q) \wedge I(s_i, s_j) > I(s_j, s_q) \wedge I(s_i, s_q) \neq I(s_j, s_q)\}$.

Group-by operator. The **group-by** operator is employed to reduce large volumes of data, and is often employed to aggregate data from several tuples into a single one. Thus, it performs two operations: (i) first groups the tuples; then (ii) aggregates data from each group. It is represented in the Relational Algebra as $\boxed{\gamma_{\{L_G, L_A\}}\mathsf{T}}$, where: T is the relation to be processed; L_G is a subset of scalar attributes E_i from T, that identify the groups. Each attribute $\mathsf{E}_i \in L_G$ is called a *grouping attribute*; L_A is a list of aggregate functions, each one applied to a scalar attribute $\mathsf{E}_i \in \mathsf{T}$. Each L_A element has the form $\boxed{f_{Aggreg}\,(\mathsf{E}_i)}$, where f_{Aggreg} is an aggregation function that summarizes the values of E_i from every tuple in a group into a single value. E_i is called an *aggregated attribute*. The result is a new relation where the attributes L_G are the primary key and the other attributes are the result of applying each function f_{Aggreg} over the corresponding aggregated attribute (E_i) from the original relation. As it processes just one input relation, group-by is a unary operator. It is executed in three steps: **(1) Identify the groups:** Perform a duplicate removal operation over attributes L_G, creating a temporary relation T_{temp} whose primary key is L_G; **(2) Assign each tuple to the respective group:** Join each tuple in the original relation to a key in T_{temp}, preparing the tuple subsets to be sent to the aggregation functions; **(3) Perform the aggregation:** Evaluate the aggregation function over each group and fill each tuple that summarizes the group.

Related Work. Extensions to the classic group-by operator have been proposed in the literature, including exploiting functionalities required by new domains [Silva et al. 2019]. In an early work on similarity-based grouping, Schallehn et al. (2004) extended SQL to allow user-defined similarity functions in the `GROUP BY` clause and in similarity grouping predicates, focusing on data integration based on string similarity predicates. However, it is restricted to a specific data type, trading attribute identity comparisons by similarity range. Other works are being developed aiming at supporting similarity operations on RDBMSs. SimDB [Silva et al. 2010] is a PostgreSQL extension that also supports similarity-based queries and is currently the only one that performs similarity-based grouping, although it is limited to work with one-dimensional data. It was later extended to multidimensional data and range similarity predicates [Tang et al. 2016].

3. Similarity Grouping by Influence

Although group-by $\gamma_{\{L_G, L_A\}}\mathsf{T}$ is a unary operator that compares attributes only by identity, it assigns two distinct roles to the input relation: first, it extracts the seeds of groups, and then, it groups the tuples comparing the seeds with the original relation, as if they were distinct (as described in Section 2). Following this rationale, and aiming at extending the group-by operator to better suit the similarity with diversification grouping, we define the Similarity Grouping by Influence operator (SGI) as a binary operator that receives a relation of seeds together with the relation to be grouped and allows representing similarity-based predicates (θ_s), as presented in Definition 1.

Definition 1 ***Similarity Grouping by Influence operator (SGI)****: The SGI is represented as a binary operator* $\mathsf{T}_1 \overset{(c(L_G))}{\gamma}_{\{L_A\}} \mathsf{T}_2$*, where:*

- $c(L_G)$ *is the* ***grouping predicate****, expressed as a logic expression on terms of* $A_{Gi}\ \theta\ A_{Gj}$*, such that* $A_{Gi} \in \mathsf{T}_1$ *and* $A_{Gj} \in \mathsf{T}_2$ *are the grouping attributes,* $Dom(A_{Gi}) = Dom(A_{Gj})$*, and* θ_s *is a comparison operator valid in that domain;*
- L_A *is the list of aggregate functions. Each* L_A *element has either the form* $f_{Aggreg}(\mathsf{E}_i)$ *or* $f_{\ddot{A}ggreg}(S_i, S_j)$*, where* f_{Aggreg} *is an aggregate function summarizing the active domain of scalar attributes and* $f_{\ddot{A}ggreg}$ *is an aggregate function summarizing by similarity the active domain of two complex attributes* S_i *and* S_j*;*
- T_1 *is the* ***seed relation****. It has a tuple for each group to be generated, and must contain at least the grouping attributes* L_G*;*
- T_2 *is the* ***grouping relation****. It must contain at least the grouping attributes* L_G *plus the attributes employed in any of the aggregate functions in* L_A*.*

The result is a relation T_R that has one tuple for each tuple of relation T_1. Relation T_R has all attributes of T_1 plus a new attribute for each element in L_A. To ensure diversity, the θ_s operator used in the grouping by influence predicate c is influence-based (section 2), which creates a ternary relationship involving the seed reference in T_1, the grouping relation T_2, and the result set T_R, to set a minimum distance that two elements must attend to be considered diverse from each other, otherwise, they should be grouped as similar. The influence criteria require only the seed reference element to group tuples.

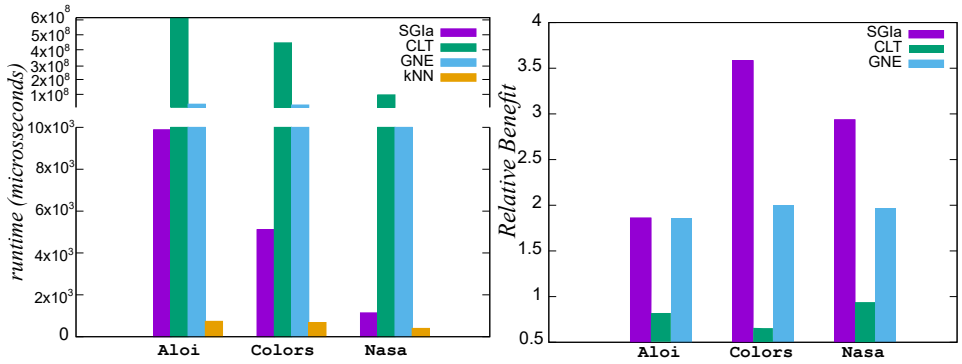
We developed a *Similarity Grouping by Influence* algorithm (SGIa), that incrementally splits the grouping relation by selecting (non-influenced) representatives to create groups of elements influenced by the seed references $s_g \in \mathsf{T}_1$. The algorithm is a nested loop strategy, which incrementally evaluates each tuple in T_2 ranked in A_g by s_g, to assert the influence set criteria. When T_2 is not empty, SGIa selects the nearest element to the group representative s_g. At each iteration, the element $s_i \in \mathsf{T}_2$ nearest to the group representative is chosen and used to retrieve the elements of the influenced set. Thus, SGIa evaluates whether s_j is within the influence set of element s_i. If $I(s_i, s_j) \geq I(s_j, s_q)$ then s_j is assumed to be influenced by s_i and is included in the group of elements influenced by s_i.

The traditional group-by can be expressed with SGI, requiring just that T_1 and T_2 are the same relation and the predicate c is a conjunction of terms compared by the $=$ operator. However, there are two benefits brought by the SGI operator. First, the θ in c can be any comparison operator, including similarity-based (θ_s) ones. Second, it enables defining new aggregate functions $f_{\ddot{A}ggreg}$ - similarity-based - on L_A.

4. Experimental Evaluation

In this section, we compare our proposed SGIa algorithm with three other approaches: (i) The baseline grouping, that is, the non-diversified k-nearest neighbor algorithm (k-NN), and (ii) a grouping algorithm based on diversity with k-medoids clustering algorithm (CLT) [van Leuken et al. 2009], (iii) a grouping algorithm based on diversification with an optimization approach (GNE) [Vieira et al. 2011].

We evaluated our proposal measuring two query properties on three image



(a) Performance Test *(a) Quality Test*

Figure 1. Experimental results.

datasets: `Aloi`[2], with $72,000$ 3D color model images and 144 dimensions; `Colors`[3], with $112,000$ low-level features from color photos with 111 dimensions; `Nasa`[3], with $40,150$ low-level features from satellite images with 20 dimensions. First, we evaluated the computational cost required to execute the grouping operator. Second, we evaluated the grouping quality, measuring the relative benefit regarding the change of retrieving the most similar answers to a group of representative elements.

For the Performance test, we measured the average wall-clock time required to evaluate 500 queries that return different values of k groups. The seed relation T_1 has 500 query elements that were randomly chosen among the grouping relation from each dataset - The results are shown in Figure 1(a). As expected, the standard grouping using k-NN is the fastest method, as it does not consider the diversity among the elements in the result set. Therefore, our comparison employs the k-NN as the baseline to measure how much a grouping with diversity method is closer to the minimum theoretical execution time. SGIa was slower than k-NN by almost two orders of magnitude, but it was always much faster than the other approaches. In fact, SGIa was consistently around five orders of magnitude faster than GNE and CLT. These results show that our approach outperforms all the competitors by a significant margin.

For the second property explored in the experiments, we compared the answer quality obtained by SGIa and its competitors using the *Relative Benefit* (RB) factor, which measures how much of the similarity is compromised when adding diversity to the result [Smyth and McClave 2001]. The rationale is that a good set of representatives should have more diversified elements than only presented by the most similar elements. The results are shown in Figure 1(b), where higher values indicate better algorithms. It shows that SGIa achieves the best performance for every dataset, with an RB average of 2.83. CLT performed poorly, with RB always less than 1 for all datasets (an average of 0.81). This happens because the cluster centroid is obtained without taking into account the distances to the query element. GNE also presented RB factors better than one, but consistently worse than SGIa, only drawing for `Aloi`. These gains are achieved because SGIa builds the result set incrementally using the representatives' influence to automatically calculate the separation distance.

[2]Amsterdam Library of Object Images: https://aloi.science.uva.nl
[3]SISAP databases: https://www.sisap.org/dbs.html

5. Conclusions

In this paper, we presented the binary Similarity Grouping by Influence (SGI) operator, based on the concept of "influence" to ensure diversity among the group elements. Besides, we also detailed its algebra to fit into an RDBMS traditional group-by. The SGIa algorithm selects the elements more dissimilar to the others to be separated under distinct "group representatives", ensuring a minimum distance separation. The results spotted that SGIa performs the grouping operation with increasing diversity in results and costs computationally less than its competitors. Future works include *(i)* performing an extensive set of experiments with different data domains and quality metrics, and *(ii)* exploring index strategies to enhance the SGIa algorithm.

Acknowledgments. The study was supported by CNPq, CAPES, and FAPESP (Grant $2016/17078-0$).

References

Drosou, M., Jagadish, H. V., Pitoura, E., and Stoyanovich, J. (2017). Diversity in big data: A review. *Big Data*, 5(2):73–84.

Jasbick, D. L., Santos, L. F. D., de Oliveira, D., and Bedo, M. V. N. (2020). Some branches may bear rotten fruits: Diversity browsing vp-trees. In *SISAP 2020*, volume 12440, pages 140–154. Springer.

Lopes, C. R., Santos, L. F. D., Jasbick, D. L., de Oliveira, D., and Bedo, M. V. N. (2021). An empirical assessment of quality metrics for diversified similarity searching. *J. Inf. Data Manag.*, 12(3).

Santos, L. F. D., Carvalho, L. O., Oliveira, W. D., Traina, A. J. M., and Jr., C. T. (2015). Diversity in similarity joins. In *SISAP 2015*, volume 9371, pages 42–53. Springer.

Schallehn, E., Sattler, K.-U., and Saake, G. (2004). Efficient similarity-based operations for data integration. *Data & Knowledge Engineering*, 48(3):361–387.

Silva, Y. N., Aly, A. M., Aref, W. G., and Larson, P.-A. (2010). SimDB: a similarity-aware database system. In *ACM SIGMOD*, pages 1243–1246. ACM.

Silva, Y. N., Sandoval, M., Prado, D., Wallace, X., and Rong, C. (2019). Similarity grouping in big data systems. In *Similarity Search and Applications*, pages 212–220.

Smyth, B. and McClave, P. (2001). Similarity vs. diversity. In *Proceedings of the ICCBR*, pages 347–361, Vancouver, Canada.

Tang, M., Tahboub, R., Aref, W., Atallah, M., Malluhi, Q., Ouzzani, M., and Silva, Y. (2016). Similarity group-by operators for multi-dimensional relational data. *Knowledge and Data Engineering, IEEE Transactions on*, 28(2):510–523.

van Leuken, R. H., Garcia, L., Olivares, X., and van Zwol, R. (2009). Visual diversification of image search results. In *Proceedings of the WWW*, pages 341–350, Spain.

Vieira, M. R., Razente, H. L., Barioni, M. C. N., Hadjieleftheriou, M., Srivastava, D., Traina Jr., C., and Tsotras, V. J. (2011). On query result diversification. In *Proceedings of the IEEE ICDE*, pages 1163–1174, Hannover, Germany.

Yang, C., Chen, L., Wang, H., Shang, S., Mao, R., and Zhang, X. (2023). Dynamic set similarity join: An update log based approach. *IEEE Trans. Knowl. Data Eng.*, 35(4):3727–3741.