Combine-and-conquer: improving the diversity in similarity search through influence sampling

Symposium on Applied Computing, 30th, 2015, Salamanca.
http://www.producao.usp.br/handle/BDPI/49015

Downloaded from: Biblioteca Digital da Produção Intelectual - BDPI, Universidade de São Paulo
Combine-and-Conquer: Improving the Diversity in Similarity Search Through Influence Sampling

Department of Computer Science, University of Sao Paulo, Brazil
{luciodb, willian, olmes, monika, agma, caetano}@icmc.usp.br

ABSTRACT
Result diversification methods are intended to retrieve elements similar to a given object whereas also enforcing a certain degree of diversity among them, aimed at improving the answer relevance. Most of the methods are based on optimization, but bearing NP-hard solutions. Diversity is injected into an otherwise all-too-similar result set in two phases: in the first, the search space is reduced to speed up finding the optimal solution, whereas in the second a trade-off between diversity and similarity over the reduced space is obtained. It is assumed that the first phase is achieved by applying a traditional nearest neighbor algorithm, but no previous investigation evaluated the impact of the first over the second phase. In this paper, we devised alternative techniques to execute the first phase and evaluated how obtaining a better quality set of elements in the first phase can improve the diversity. Besides the traditional nearest neighbor-based pre-selection, we also considered naive random selection, cluster-based and influence-based ones. Thereafter, extensive experiments evaluated a number of state-of-the-art diversity algorithms employed in the second phase, regarding both processing time and answer quality. The obtained results have shown that although the much more elaborated (and much more time consuming) methods indeed provide best answers, other alternatives are able to provide a better commitment regarding quality and performance. Moreover, the pre-selection techniques can reduce the total running time by up to two orders of magnitude.

Categories and Subject Descriptors
H.2.4 [Database Management]: Systems—Query processing

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Copyright 2015 ACM 978-1-4503-3196-8/15/04$15.00.
http://dx.doi.org/10.1145/2695664.2695798

1. INTRODUCTION
With the increasing ability of online applications to produce multimedia data, it has become more and more important to provide improved techniques to find relevant information and to intuitively present them to the users. Until now, similarity-based searching seemed suitable for the retrieval task. However, for applications handling a massive database, “pure” similarity generally produce answer sets whose elements are indeed similar to the query center, but also too much similar among themselves, leading to almost-repeated elements that do not aggregate new information.

In such context, several researchers from different domains have explored result diversification as a way to obtain results that include elements not only similar to the query center, but also with a certain diversity among the returned elements [1, 2, 4]. The most common way to induce diversity into similarity search is solving a bi-criteria optimization problem. In this approach, similarity and diversity compete with each other, ruled by a user defined trade-off parameter. However, optimal solutions for this approach have worst-case NP-hard computation time [6, 5].

To speed up execution, result diversification methods typically reduce the search space including an initial step to filter a subset of the original dataset. Thus, a k-nearest neighbor query with diversity (k − DNNq) obtains the final k elements in two phases: candidate filtering and diversity computation. In general, the first phase is performed applying a traditional m-nearest neighbor algorithm to retrieve a candidate set S such that |S| = m and m > k. The diversity computation phase processes just S to select the final diversity set R ⊆ S, |R| = k maximizing a bi-criteria objective function that balances similarity and diversity. Both the number m of elements and how they are selected play important roles in this phase. As computing diversity requires at least m^2 element-to-element comparisons, low-cardinality candidate sets must be employed [6, 5].

Most of the previous researches focus only on the diversity computation phase, with the majority of them exploring greedy heuristics with different construction strategies to obtain the diversity set [1, 2, 6]. To the best of our knowledge, none of the previous researches investigated the impact of the candidate filtering phase, regarding neither the efficacy nor the efficiency that could be achieved by replacing the traditional nearest neighbor algorithm.

In this paper, we explore how the main existing diversifica-

Keywords
Search result diversification, similarity search, sampling
tion algorithms benefit from changing the filtering phase to obtain better quality candidate sets. We show that employing improved techniques to reduce the number of candidates allows the diversity algorithms to execute over much larger datasets in feasible time without compromising the final answer quality. Practical evaluation showed that it may reduce the total running time by up to 2 orders of magnitude.

The remainder of this paper is structured as follows: Section 2 presents the main related concepts. Section 3 describes the methodology adopted. Section 4 details the experiments performed over two real datasets and analyzes the results achieved. Finally, Section 5 summarizes the conclusions of this work.

2. BACKGROUND

A similarity query retrieves the elements most similar to an element given as the query center, using a distance function to compute how similar two elements are. The two most common similarity-based operators are the $k$-nearest neighbor ($k$-NN) and the similarity range ($R_q$). While a $k$-NN retrieves the $k$ elements most similar to the query center, the $R_q$ retrieves the elements distant at most a given radius $\xi$ from the query center. The main concern regarding those operators is that they may get results containing several elements too similar to themselves: they consider only the similarity between each element and the query center, and do not take into account the “diversity” among the elements themselves.

2.1 The Diversity Definition

The diversity problem can be stated as “how to retrieve elements similar to the query center, but also diverse enough to generate a more heterogeneous and useful result set” [4]. Let $D = \{d_1, ..., d_n\}$ be the set of $n$ stored elements, $d_q$ a query center and $k \leq n$ an integer. Let also the similarity between any $d_i \in D$, $1 \leq i \leq n$ to $d_q$ be specified by the distance function $\delta_{sim} : d_q \times D \rightarrow \mathbb{R}^+$ and the diversity between pairs of elements $d_i, d_j \in D$, $i \neq j$, defined by the distance function $\delta_{div} : D \times D \rightarrow \mathbb{R}^+$. Let $\lambda \in [0, 1]$ be a trade-off parameter that specifies the proportion between similarity and diversity, called here as the diversity preference. The diversity problem requires to compute the result set $R$ according to Equation 1, following.

$$R = \arg \max_{\mathcal{R}} \mathcal{F}(d_q, \mathcal{R}), \forall \mathcal{R} \subseteq D \text{ } k = |R|,$$

where

$$\mathcal{F}(d_q, \mathcal{R}) = (k - 1)(1 - \lambda) \cdot \text{Sim}(d_q, R) + 2\lambda \cdot \text{Div}(R),$$

$$\text{Sim}(d_q, R) = \sum_{i=1}^{k} \delta_{sim}(d_q, d_i), d_i \in R \text{ and }$$

$$\text{Div}(R) = \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \delta_{div}(d_i, d_j), d_i, d_j \in R.$$ (4)

Drosou and Pitoura [2] showed that the user tends to prefer result sets where the diversity preference is around 0.4 when compared to results that only similarity ($\lambda = 0$) is considered, and also that higher diversity ($\lambda > 0.5$) leads to a smaller users interest.

The optimal solution for the diversity problem with diversity preference $0 < \lambda < 1$ is gotten by exhaustively verifying each possible answer set $R$, $|R| = k$, and selecting the one with the highest $\mathcal{F}$. Assuming $R \subseteq D$ with cardinality $|R| = k$ and $|D| = n$, the total number of subsets composed of $k$ elements is given by the combination $C(n, k)$, resulting in time complexity $O(n^k)$. In addition, each subset requires $O(k^2)$ distance calculations to evaluate the objective function $\mathcal{F}$, resulting in an overall time complexity $O(n^k k^2)$. Therefore, in real situations using large databases, the exhaustive approach to compute diversity becomes prohibitive.

Even employing sub-optimal algorithms to obtain approximate answers faster, it remains interesting to filter the original dataset $D$ in order to obtain a candidate subset $S \subseteq D$ that can lead to high objective function values. Using this approach, the diversity problem is performed in two phases: first pick the candidate elements in order to reduce the complexity and then compute the diversity over them. Traditionally, the former often employs a $k$ nearest neighbor selection ($k$NN) so that elements with small relevance to the query may be excluded at once [1, 2, 3, 4, 6, 7]. Those candidate elements are used as input for the latter phase in order to compute the diversity set.

2.2 The Diversity Computation

The computation of the diversity set is based on greedy algorithms [6, 2]. The existing greedy heuristics are classified according to the construction strategy as incremental, exchanging or as meta-heuristic.

The algorithms following the incremental strategy start with the result set $R$ empty and iteratively select candidates that maximizes the objective function $\mathcal{F}$. The Maximal Marginal Relevance (MMR) [1], Greedy Marginal Contribution (GMC) [6] and Max-Sum Dispersion (MSD) [3] are examples of algorithms that follow the incremental strategy. The MMR is the fastest among them, while GMC provides the best results (higher quality) [4].

The algorithms following the exchanging strategy initialize the result set $R$ with the $k$-nearest neighbors computed in the candidate filtering phase. Thereafter, the remaining candidates are evaluated aiming at replacing an element in the current solution $R$. Swap [7] is an algorithm that employs this strategy. However, Swap is outperformed by any greedy algorithms following the incremental strategy [4].

The algorithms following the meta-heuristic strategy create an initial result set $R$ using a greedy randomized ranking function that ranks the candidate elements according to each individual contribution to the objective function. Thereafter, a local search iteratively improves the current solution $R$ swapping an element in the result set. The Greedy Randomized with Neighbor Expansion (GNE) is an example of this strategy. It presents the best quality among all the others greedy algorithms, but it is also the slowest among them all [6].

Following we detail the main algorithms that we employed as the second phase in our experimental evaluation.

Maximal Marginal Relevance (MMR)

The MMR algorithm iteratively constructs the result set $R$ selecting a new element $s_i \in S, S \subseteq D$, that maximizes the objective function (Equation 5):

$$\mathcal{F}_{\text{MMR}}(s_i) = (1 - \lambda)\delta_{sim}(d_q, s_i) + 2\lambda \sum_{s_j \in R} \delta_{div}(s_i, s_j).$$ (5)

That algorithm starts with the candidate with the smallest $\delta_{sim}$ to $d_q$, regardless of the value of $\lambda$, i.e., the most similar element to the query center. Then, the result is incrementally built by inserting new elements selected among those
with the highest value of $F_{MMR}$. Notice that the first element has a large influence in the final result set $R$ quality, since it is chosen without considering diversity.

**Greedy Marginal Contribution (GMC)**

The GMC algorithm is very similar to the MMR, as it also incrementally select the elements with the largest contribution to the answer. However, GMC uses a different objective function, called the maximum marginal contribution (mmc), defined according to Equation 6.

$$mmc(s_i) = (1 - \lambda)\delta_{sim}(s_i, d_q) + \frac{\lambda}{k - 1} \sum_{s_j \in S - s_i} \delta_{div}(s_i, s_j),$$

(6)

The mmc function considers not only the similarity ($\delta_{sim}$) between a candidate $s_i$ and $d_q$, and the diversity among the elements already in the result set $R_{t-1}$ ($\delta_{div}$), but also the diversity among those that were not included in the result set, $s_j \in S - s_i$ ($\delta_{div}$).

**Max-Sum Dispersion (MSD)**

The MSD algorithm computes the result set $R$ by incrementally picking element pairs that are both similar to the query center $d_q$ and diverse among themselves. In each iteration, it chooses two elements $s_i, s_j \in S$ that maximize the objective function given by Equation 7.

$$F_{MSD}(s_i, s_j) = (1 - \lambda)(\delta_{sim}(d_q, s_i) + \delta_{sim}(d_q, s_j)) + 2\delta_{div}(s_i, s_j).$$

(7)

As this objective function evaluates element pairs, when $k$ is odd, the method selects randomly one element in $S$ for the last iteration. An inherent problem with MSD is to consider the diversity between pairs of elements without checking whether the insertion of the new pair includes elements too close to the elements already in the result set.

**Swap**

The Swap algorithm is executed in two steps. First, the $k$ elements in $S$ nearest to $d_q$ create the initial result $R$. Then, each element remaining in $S$ (ordered by decreasing similarity $\delta_{sim}$ to $d_q$) is evaluated to replace an element in the current solution $R$. Whenever an element improves $F_R$, then an exchange operation occurs until every element in the candidate set $S$ is evaluated. The final result set may not be optimal, since the candidate set $S$ is analyzed with respect to the $\delta_{sim}$ ordering and does not consider the order of $\delta_{div}$ values among the elements.

**Greedy Randomized with Neighbor Expansion (GNE)**

The GNE algorithm is performed in two phases: construction and local search. In each construction iteration, the choice of the next element to be added in $R$ is determined by a greedy randomized ranking function, which ranks the elements in $S$ following Equation 6. Thus, only the elements with the highest individual $mmc$ contribution are considered to be stored in a so called Restricted Candidate List (RCL). Thereafter, an initial result set $R$ is randomly chosen from RCL, which may not have the elements with the highest contribution.

The second phase executes a local search algorithm, progressively improving the initial result by applying a sequence of local modifications in the neighborhood of the current solution. It exchanges elements in the result set $R$ for the most diverse elements with respect to a reference element in $R$, whenever this new element improves the current solution. Although GNE has the higher quality among all the others greedy algorithm, it is the slowest among them [6].

**2.3 Result Diversification Based on Influence**

The Result Diversification based on Influence (RDI) is a recent approach that defines diversity using the separation distance principle [5]. This technique assumes that if two elements ($d_i$ and $d_j$) are closer than a minimum distance, they ought to bring the same amount of information and only one of them should be returned. Such minimum distance is estimated using the concept of “influence” intensity ($I$), defined as the inverse of the similarity distance ($\delta_{sim}$) between $d_i$ and $d_j$. Let $d_i, d_j$ and $d_q$ be elements in $D$. Then $d_j$ is assumed to be more influenced by $d_i$ than by $d_q$ if $I(d_i, d_j) \geq I(d_j, d_q)$. For a query center $d_q$, the RDI goal is to retrieve a diversity result set $R \subseteq D$ by selecting elements in $D$ that are similar to $d_q$, but also considering the minimum distance between two elements $d_i, d_j \in R$ by the influence intensity $I$.

The BRID (Better Result with Influence Diversification) [5] technique implements the influence concept for $k$-$NNq$ and $Rq$ similarity comparison operators. This technique incrementally builds the result set by selecting the element most similar to $d_q$ and, for each iteration, checks the influence intensity between the elements, returning only those elements relevant to the query (either nearest or in the range) that are not influenced by others.

**3. METHODOLOGY**

The candidate filtering phase aims at reducing the complexity of the exhaustive solution by pruning the original dataset, so that elements with small relevance to the query may be excluded at once. As aforestated, the methods studied in the literature restrict the search space to the nearest elements ($k$-$NN$). The goal of our proposal is to reduce the number of elements by selecting candidates with improved diversity, ensuring that similarity is still preserved in the answer, but avoiding elements that bring a too low contribution to the diversity computation phase.

**3.1 Techniques to improve the candidate filtering phase**

We devised alternative techniques to generate the candidate set, including selecting elements at random, using clustering algorithms, such as a $k$-medoid, and selecting elements based on the influence concept. Each technique has an underline assumption to improve the quality and/or performance of diversity algorithms.

**Random selection method (Rnd)**

The simplest technique to generate a candidate set is the random selection of elements in $D$. However, the elements selected must be relevant to the query, thus it is required to prune the search space by assigning a maximum similarity distance $\xi_{max}$ from $d_q$. Therefore, random selection (Rnd) randomly chooses elements $s_i \in D | \delta_{sim}(d_q, s_i) > \xi_{max}$ until enough elements are pre-selected. This naive approach is faster than any other algorithm, as it just requires one distance evaluation for each element selected, without fur-
ther analysis. It requires two parameters: the number $p$ of elements to be randomly selected and the maximum distance threshold $\xi_{max}$ from each element to the query center. Throwing a correct definition for parameter $p$ in general requires several trial executions varying its value until a suitable balance among quality and performance is achieved. Although $Rnd$ reduces the time spent in the candidate selection phase, there is no guarantee that it generates a good quality candidate set.

**Clustering-based method (CLT)**

Using a clustering algorithm certainly requires more processing than either the random or nearest neighbor approaches. However, the reasoning is that clustering may generate candidate sets that better summarizes the neighborhood around the query center. It can be employed in two ways. The first way groups the entire dataset without restricting the search space to the elements most similar to the query. The second is to group the elements already filtered by a previous $kNN$ step. The later approach is faster than the former, since the cardinality of the filtered set may be much smaller than the cardinality of the entire dataset, without imposing significant adverse effects on the result. Therefore, in our experiments we evaluated the clustering approach using the $k$-medoid algorithm on the result of a $kNN$ filter. The number $k$ of groups in the $k$-medoid should be defined a priori, which also requires executing the clustering algorithm several times to find out how many groups might exist.

**Influence selection (BRID)**

The influence selection keeps the fundamental nature of selecting similar elements with a minimum distance among them. Thus, the candidate set may be generated only with good diversity candidates with respect to the query center. In this approach, we propose to consider the diversity starting from the first phase, as the BRID algorithm is based on the separation distance principle applied over the entire dataset. Thus, this approach divides the computational cost of processing diversity by combining two different diversity definitions, allowing the second phase to act as a diversity refinement phase. Unlike the methods aforementioned, BRID does not require defining the number of candidate elements nor any other parameter.

### 3.2 Evaluation strategy

We followed two strategies to evaluate the impact of the candidate filtering phase: (i) measure the time demanded to execute the diversity queries over candidate sets generated by the $kNN$, $Rnd$, CLT and BRID methods; (ii) measure the quality of the diversity-enabled result sets with respect to the objective function.

The evaluation of the objective function [4] measures the maximization of the result sets based on the diversity function defined for each method. To have a standard reference for comparison, we applied the results of all the evaluated algorithms to the objective function shown in Equation 1. For instance, considering that any two algorithms $A$ and $B$ were defined, then algorithm $B$ is considered better than $A$ if $F_B$ is higher than $F_A$, regardless of different strategies for candidate filtering. Figure 1 shows our proposed framework. We also included the traditional $kNN$ algorithm, in order to allow combining any candidate filtering strategies prior to the diversity computation phase.

![Figure 1: The framework overview.](image)

### 4. EXPERIMENTS

This section presents the experimental results on the comparisons performed using the candidate filtering techniques (Section 3.1) before executing the diversity algorithms (Section 2.2). The experiments aim at evaluating the trade-off among performance and quality of the filtering step when applying different strategies to generate candidate sets.

#### 4.1 The Dataset Descriptions

Due to space limitations, in this paper we only report the experiments performed on two real datasets: U.SCities and Alo. Experiments performed over several other datasets presented similar results. The U.SCities dataset consists of geographical coordinates and economic characteristics of 25,375 American cities, obtained from the U.S. Census Bureau website\(^1\). The elements were compared using the Euclidean distance ($L_2$) for both $\delta_{sim}$ and $\delta_{div}$ over the latitude and longitude coordinates. The Alo dataset consists of 72,000 color images rotated in 5 degree steps, obtained from the Amsterdam Library of Object Images website\(^2\). The feature vector of each image was extracted using the color moment extractor, containing 144 features. In this dataset we used the Euclidean distance ($L_2$) for both $\delta_{sim}$ and $\delta_{div}$.

The experiments were performed on a computer built with an Intel\(^\text{®}\) Core\(^\text{TM}\) i7 processor, with 8 GB of RAM, running the operating system GNU Linux distribution Ubuntu 11.10. All algorithms were implemented in C++ using the same programming framework to enable fair comparisons.

#### 4.2 Performance Experiment

We evaluated the performance impact that each candidate filtering approach poses on each of the algorithms reviewed in Section 2.2. For each evaluated dataset, we randomly chose 100 distinct elements as query centers and fixed the number of diversity elements retrieved in $k = 5$. Figures 2 (a), (b), (c), (e), (f) and (g) measured the average time in microseconds required by the diversity computation phase when the search space varies from 400 to 2,000 in steps of 800 elements. For instance, in a search space with 400 elements, all the candidate filtering methods generated a candidate set with 400 elements based on the query center. As the CLT and $Rnd$ methods require a parameter $p$ to define

---


The number of groups and of random selected elements for each query, we performed the same queries for values of $p$ ranging from 25% to 75% of the search space in steps of 10%, and selected the value with the best trade-off between quality and performance. For all experiments it was found that the value of $p$ was $\{100, 300, 500\}$ for $CLT$ and $400, 1200$ and $2000$ for $Rnd$.

Figures 2(a) to 2(d) show the performance results for the USCities dataset. As it can be seen in Figure 2(a), except for $MMR$, all algorithms were faster using any one of our proposed alternative candidate filtering methods than when using the traditional $kNN$ approach. For example, to execute GNE in the diversity computation phase, the $BRID$ approach reduced up to 16 times the running time required by $GNE$. Interestingly, $MSD$ is the slowest algorithm when using the traditional $kNN$ approach, but it became the fastest one when the selection phase is performed by the $BRID$ approach.

Figures 2(b) and 2(c) shows that, as the search space increases, every algorithms benefit from a better filtering approach. Notice that $BRID$ was the filtering method that most improved the diversity computation phase for every algorithm. This is due to the fact that this strategy automatically defines the number of representative elements that are not influenced by each other.

Figure 2(d) shows the total running time (including the candidate filtering phase and the diversity computation phase) when the search space is restricted to 1,200 elements. The $CLT$ approach increased the total time in 6 times when compared to $kNN$, showing that using a costly algorithm in the first phase may reduce the diversity computation phase of the algorithms but it is not enough to decrease the entire query processing time. Moreover, despite the $Rnd$ be faster than $BRID$ in the filtering phase, it required executing several queries to find the adequate number of elements in the candidate set, which anyway was always larger than those obtained by $BRID$.

Figures 2(e) to 2(h) show the performance results for the Aloi dataset. This experiment aimed at evaluating the candidate filtering approaches over high-dimensional data. As it can be seen in Figures 2(e) to 2(g), every algorithm benefits from improved candidate filtering strategies, regardless of the search space size. Figure 2(h) shows that the total running time for $CLT$ becomes prohibitive. Still in Figure 2(h), we can see that $GNE$, known as the method resulting in the highest quality, was also the most benefited from the $BRID$ phase, and now it runs in a time closer to that of $MMR$, which was the fastest one.

The results presented in this section pinpoint that a better filtering approach can improve the performance of the algorithms when compared to the traditional $kNN$ strategy. The combination of the influence sampling obtained by $BRID$ allows decreasing the running time of the slowest algorithm ($GNE$) in up to 3 orders of magnitude, while $Rnd$ decrease the running time of the fastest one ($MMR$) in almost 10%.

### 4.3 Quality Experiment

In order to evaluate the quality of the final answers of the diversity algorithms, we measured the objective function $F$ using the traditional $kNN$ and the other candidate filtering approaches. For each dataset, we randomly chose 100 different elements to be employed as query centers and set the number of diversity elements retrieved as $k = 5$ (due to the lack of space we only present the results for search space with 1,200 elements). We varied the diversity preference ($\lambda$) from 0.1 (mild diversity) to 0.5 (balanced similarity/diversity) in steps of 0.2.

Figure 3(a) to 3(c) show the quality results for the USCities dataset. As it can be seen, every filtering approach maintains the same quality provided by the traditional $kNN$. It shows that although the number of elements selected by $BRID$ is on average 6.8% (78 elements) of the number selected from $kNN$ (1,200), the $BRID$ approach can improve the performance without compromising the quality of the answer. Figure 3(c) shows the results for a balanced-diversity preference ($\lambda = 0.5$), where the highest gap among the filtering approaches related to $kNN$ was up to 2% ($BRID$),
while CLT and Rnd had almost the same value achieved by the kNN (less than 1%).

Figures 3(d) and 3(e) show that only the BRID approach achieves the same quality of kNN. Notice that the Rnd approach achieved the lower quality, although it loosened only by about 1%. Moreover, every filtering strategy had approximately the same quality for balanced-diversity preference (Figure 3(f)).

5. CONCLUSIONS

The similarity operators are the most often employed to process queries over multimedia data. However, in large databases, similarity-based queries often retrieve result containing elements too much similar among themselves, which does not add much valuable information to the query.

In such context, result diversification provide a promising solution, making it possible to retrieve elements similar to the query center yet diverse among themselves. Traditionally, diversity methods are composed by two steps: candidate filtering and diversity computation phases. Although many efforts were put in the second phase, previous works always employed a k-nearest neighbor approach to filter the candidates for the diversity computation.

In this paper we showed the importance of the candidate filtering phase to the main algorithms existing in the literature, regarding both the performance and quality of the final answer. Extensive empirical evaluation shows that obtaining a better quality subset in the candidate filtering phase contributes to reduce the number of candidates and allows executing diversity algorithms over large datasets in a feasible time, without compromising the final answer quality.

To validate our methodology, we performed experiments using two real datasets that span up to 70,000 data elements and are represented by feature vectors of up to 144 dimensions. Alongside the kNN, we evaluated three candidate filtering strategies to compute the candidate set: BRID, Rnd and CLT. The experiments revealed that the performance of the diversity algorithms quality improve when the candidate filtering phase selects only the most representative elements, reducing the number of comparisons performed in the diversity computation.

An interesting result obtained is that the best but also slowest GNE algorithm had the running time decreased by at least 2 orders of magnitude when compared to its original version. Thus, the method that had the best quality but was also the most time-consuming, when put forward to process the selection extracted by BRID is now able to maintain the quality whereas processing almost as fast as the fastest one.

6. REFERENCES