

Proximity measures for rank join

DAVIDE MARTINENGI, Politecnico di Milano
MARCO TAGLIASACCHI, Politecnico di Milano

We introduce the proximity rank join problem, where we are given a set of relations whose tuples are equipped with a score and a real-valued feature vector. Given a target feature vector, the goal is to return the K combinations of tuples with high scores that are as close as possible to the target and to each other, according to some notion of distance or dissimilarity. The setting closely resembles that of traditional rank join, but the geometry of the vector space plays a distinctive role in the computation of the overall score of a combination. Also, the input relations typically return their results either by distance from the target or by score. Because of these aspects, it turns out that traditional rank join algorithms, such as the well-known *HRJN*, have shortcomings in solving the proximity rank join problem, as they may read more input than needed. To overcome this weakness, we define a tight bound (used as a stopping criterion) that guarantees instance optimality, i.e., an I/O cost is achieved that is always within a constant factor of optimal. The tight bound can also be used to drive an adaptive pulling strategy, deciding at each step which relation to access next. For practically relevant classes of problems, we show how to compute the tight bound efficiently. An extensive experimental study validates our results and demonstrates significant gains over existing solutions.

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

General Terms: Design, Algorithms, Experimentation, Performance

Additional Key Words and Phrases: Top-k, rank-aware processing, proximity

ACM Reference Format:

Martinenghi, D., Tagliasacchi, M., 2011. Proximity Rank Join. *ACM Trans. Datab. Syst.* V, N, Article A (January YYYY), 45 pages.

DOI = 10.1145/0000000.0000000 <http://doi.acm.org/10.1145/0000000.0000000>

1. INTRODUCTION

Imagine a smartphone user wishing to organize the evening by finding a restaurant, a movie theater and a hotel that are nearby, close to each other, and recommended in terms of, respectively, price, user rating, and number of stars. This can be done by suitably collecting and assembling different pieces of information. Here, one could, e.g., exploit the current geographical position available on the smartphone, as well as local information obtained from specific search services on the Internet (such as Yahoo! Local, IMDB, etc.).

This simple example captures the essence of *proximity rank join* problems, in which the best combinations of objects coming from different services (here: restaurants, the-

This is a preliminary release of an article accepted by ACM Transactions on Database Systems. The definitive version is currently in production at ACM and, when released, will supersede this version.

The authors acknowledge support from the Search Computing (SeCo) project, funded by the European Research Council.

Author's address: Dipartimento di Elettronica e Informazione, Politecnico di Milano, Piazza Leonardo da Vinci, 32 – 20133 Milano, Italy; email: {martinen, tagliasa}@elet.polimi.it.

Permission to make digital or hard copies of part or all 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 show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.

© YYYY ACM 0362-5915/YYYY/01-ARTA \$10.00

DOI 10.1145/0000000.0000000 <http://doi.acm.org/10.1145/0000000.0000000>

aters, and hotels) are sought, and each object is equipped with both a score and a real-valued feature vector. The aggregation function assigning a score to a combination depends on the individual scores, on the proximity of the individual vectors to a given vector (called query) and on their mutual proximity, according to some notion of distance or dissimilarity in the feature space. Typically, objects are retrieved sorted by either distance¹ from a given vector or by score.

The interest in proximity rank join is motivated by its generality and broad applicability. Indeed, it covers multi-domain search, as in the previous example, as well as several other scientific fields, such as: *i*) information retrieval, e.g., finding similar documents in different data collections given a set of keywords; *ii*) multimedia databases, e.g., requesting similar images from different repositories given a sample image; *iii*) bioinformatics, e.g., discovering orthologous genes from different organisms given a target annotation profile; and many others. In all these cases, proximity (typically based on Euclidean distance or cosine similarity) plays a crucial role, as it captures the mutual relationships between the objects in the feature space.

In the conventional rank join problem [Ilyas et al. 2008], instead, the overall score of a combination depends only on the scores of the single objects. Existing algorithms, such as *HRJN* [Ilyas et al. 2004], may be used to address proximity rank join. However, as will be shown in this paper, they fail to achieve optimality as they are completely oblivious of the proximity aspect.

Our contributions. We formally define the proximity rank join problem and propose algorithms that solve it. We identify the main differences with traditional rank join and revisit existing algorithms and templates in the new setting in order to assess their limitations and potential. To this end, as customary in top-k query answering, we refer to the notion of instance-optimality [Fagin et al. 2003] to characterize I/O efficiency. We show that, for the proximity rank join problem, no algorithm using a stopping criterion based on the so-called corner bound (such as *HRJN* and *HRJN**) is instance-optimal, even with only two input relations.

We then introduce a different bound that is tight [Schnaitter and Polyzotis 2008] and can thus be used in order to guarantee instance-optimality. We show how to compute the tight bound efficiently for practically relevant instances of the problem, in particular when the proximity measure in use is based on either Euclidean distance or cosine similarity.

In order to further improve efficiency in terms of I/O as well as CPU cost, we refine the exploration strategy used by the algorithm to pull tuples from the inputs. Also, we exploit geometrical properties to conclude that part of the search space is dominated and thus need not be explored.

Finally, we provide a thorough experimental evaluation of our approach by analyzing performance with regard to the main operating parameters. These include the number of desired top combinations, the number of dimensions of the feature space, the density of tuples in the space, its skewness, and the number of relations in the join. Our empirical study is carried out both on real and on synthetic data.

A preliminary version of this work was published in [Martinenghi and Tagliasacchi 2010]. While that short version focuses on Euclidean distance, in the current article we also provide efficient solutions for rank join under cosine similarity, both with distance-based and score-based access. In addition, we explore a new, more powerful notion of dominance that applies to all cases at hand. Finally, we include the extra empirical results corresponding to the new solutions, and further deepen our analysis by consid-

¹With a slight abuse of terminology, from now on we shall use the word “distance” also when the underlying dissimilarity measure is not a metric, as when using cosine similarity.

ering new parameters for our experiments, including spatial skewness, join selectivity, and weights in the aggregation functions.

2. PRELIMINARIES

Consider a set of relations R_1, \dots, R_n where each tuple $\tau_i \in R_i$ is composed of named attributes, a real-valued feature vector $\mathbf{x}(\tau_i) \in \mathbb{R}^d$, and a score $\sigma(\tau_i) \in \mathbb{R}$. Let $\mathbf{q} \in \mathbb{R}^d$ denote a constant vector representing the query, and $\delta(\mathbf{x}(\tau), \mathbf{q})$ a distance measure between vectors $\mathbf{x}(\tau)$ and \mathbf{q} . We consider the two most common access kinds that arise in practice:

- A. *Distance-based access*: The relations R_1, \dots, R_n are accessed sequentially in increasing order of $\delta(\cdot, \mathbf{q})$.
- B. *Score-based access*: The relations R_1, \dots, R_n are accessed sequentially in decreasing order of $\sigma(\cdot)$.

We indicate with $\tau_i^{(r_i)} = R_i[r_i]$ the r_i -th tuple extracted from R_i according to the available access kind, and with $P_i \subseteq R_i$ the ordered relation containing the tuples already extracted from R_i . The number $p_i = |P_i|$ of such tuples is called *depth*.

Let $\tau = \tau_1 \times \dots \times \tau_n \in R_1 \times \dots \times R_n$ denote an element of the cross-product of the n relations, hereafter called *combination*. The *aggregate score* $\mathcal{S}(\tau)$ of τ is defined as

$$\mathcal{S}(\tau) = f(\mathcal{S}(\tau_1), \dots, \mathcal{S}(\tau_n)) \quad (1)$$

where

$$\mathcal{S}(\tau_i) = g_i(\sigma(\tau_i), \delta(\mathbf{x}(\tau_i), \mathbf{q}), \delta(\mathbf{x}(\tau_i), \boldsymbol{\mu}(\tau))), \quad (2)$$

and

- The function $f(x_1, \dots, x_n) : \mathbb{R}^n \rightarrow \mathbb{R}$ is monotonically non-decreasing in all of its arguments.
- The functions $g_i(x, y, z) : \mathbb{R} \times \mathbb{R}^+ \times \mathbb{R}^+ \rightarrow \mathbb{R}$, for $i = 1, \dots, n$, are monotonically non-decreasing in x and monotonically non-increasing in y and z .
- The vector $\boldsymbol{\mu}(\tau) \in \mathbb{R}^d$ denotes the *centroid* of a combination, defined as $\arg \min_{\boldsymbol{\omega}} \sum_{i=1}^n \delta(\mathbf{x}(\tau_i), \boldsymbol{\omega})$.

We refer to the functions g_i as the *proximity weighting functions* since they compute the *proximity weighted score* $\mathcal{S}(\tau_i)$ of the tuple τ_i participating in the combination τ . Intuitively, the proximity weighted score increases with the score and decreases with the distance from the query vector \mathbf{q} and from the combination centroid $\boldsymbol{\mu}(\tau)$. Therefore, the top combinations are those whose constituent tuples: *i*) have high scores; *ii*) are close to the query vector; *iii*) are close to each other. We note that (1) generalizes the *Maximize-Over-Location* scoring function recently defined in [Thonangi et al. 2009].

Example 2.1. As a concrete example, we consider

$$\mathcal{S}(\tau) = \sum_{i=1}^n w_s \ln(\sigma(\tau_i)) - w_q \|\mathbf{x}(\tau_i) - \mathbf{q}\|^2 - w_\mu \|\mathbf{x}(\tau_i) - \boldsymbol{\mu}(\tau)\|^2 \quad (3)$$

which is obtained by setting $f(x_1, \dots, x_n) = \sum_{i=1}^n x_i$ and $g_i(x, y, z) = w_s \ln(x) - w_q y^2 - w_\mu z^2$, i.e., adopting a Euclidean distance. The centroid $\boldsymbol{\mu}(\tau)$ can be computed as $\boldsymbol{\mu}(\tau) = \frac{1}{n} \sum_{i=1}^n \mathbf{x}(\tau_i)$. The weights w_s, w_q and $w_\mu \in \mathbb{R}^+$ can be used to tune the proximity weighting functions based on user's preferences. Note that if the scores $\sigma(\tau_i) \in [0, 1]$ then $\mathcal{S}(\tau) \in (-\infty, 0]$. Alternatively, one might take $e^{\mathcal{S}(\tau)}$ to obtain an aggregate score in the range $[0, 1]$. Table I illustrates an example with three relations, when two tuples have been retrieved from each of them, and tuples are sorted according to the distance from the query $\mathbf{q} = \mathbf{0}$. Figure 1(a) shows the location $\mathbf{x}(\tau_i)$ of the tuples in \mathbb{R}^2 represented as discs whose radius is proportional to the score (blue for R_1 , red for R_2 , green for

Table I. Three relations with distance-based access and the formed combinations with their scores.

R_1			R_2			R_3			$\tau = \tau_1 \times \tau_2 \times \tau_3$	$S(\tau)$
	σ_1	\mathbf{x}_1^T		σ_2	\mathbf{x}_2^T		σ_3	\mathbf{x}_3^T		
$\tau_1^{(1)}$	0.5	[0, -0.5]	$\tau_2^{(1)}$	1.0	[1, 1]	$\tau_3^{(1)}$	1.0	[-1, 1]	$\tau_1^{(2)} \times \tau_2^{(1)} \times \tau_3^{(1)}$	-7.0
$\tau_1^{(2)}$	1.0	[0, 1]	$\tau_2^{(2)}$	0.8	[-2, 2]	$\tau_3^{(2)}$	0.4	[-2, -2]	$\tau_1^{(1)} \times \tau_2^{(1)} \times \tau_3^{(1)}$	-8.4
...	$\tau_1^{(2)} \times \tau_2^{(2)} \times \tau_3^{(1)}$	-13.9
									$\tau_1^{(1)} \times \tau_2^{(2)} \times \tau_3^{(1)}$	-16.3
									$\tau_1^{(1)} \times \tau_2^{(1)} \times \tau_3^{(2)}$	-21.0
									$\tau_1^{(2)} \times \tau_2^{(1)} \times \tau_3^{(2)}$	-22.6
									$\tau_1^{(1)} \times \tau_2^{(2)} \times \tau_3^{(2)}$	-28.9
									$\tau_1^{(2)} \times \tau_2^{(2)} \times \tau_3^{(2)}$	-29.5

R_3). Figure 1(a) also shows three circles, whose radius is equal to the distance of the last accessed tuple from each relation. The cross-product consists of 8 combinations, also reported in Table I, sorted by their aggregate score computed using (3), by setting $w_s = w_q = w_\mu = 1$.

Example 2.2. As another concrete example, consider

$$S(\tau) = \sum_{i=1}^n w_s \sigma(\tau_i) + w_q \frac{\mathbf{q}^T \mathbf{x}(\tau_i)}{\|\mathbf{q}\|^{\|\mathbf{x}(\tau_i)\|}} + w_\mu \frac{\boldsymbol{\mu}(\tau)^T \mathbf{x}(\tau_i)}{\|\boldsymbol{\mu}(\tau)\| \|\mathbf{x}(\tau_i)\|} \quad (4)$$

which is obtained from (1) and (2) by setting $f(x_1, \dots, x_n) = \sum_{i=1}^n x_i - 2n$ and $g_i(x, y, z) = w_s x + w_q y + w_\mu z$. Here, we have adopted cosine similarity between feature vectors, i.e., we pose $\delta(\mathbf{x}(\tau_i), \mathbf{q}) = 1 - cs(\mathbf{q}, \mathbf{x}(\tau_i))$ and $\delta(\mathbf{x}(\tau_i), \boldsymbol{\mu}) = 1 - cs(\boldsymbol{\mu}, \mathbf{x}(\tau_i))$, where cs indicates cosine similarity and is computed as follows:

$$cs(\mathbf{x}(\tau_i), \mathbf{x}(\tau_j)) = \frac{\mathbf{x}(\tau_i)^T \mathbf{x}(\tau_j)}{\|\mathbf{x}(\tau_i)\| \|\mathbf{x}(\tau_j)\|} \quad (5)$$

The centroid $\boldsymbol{\mu}(\tau)$ can be computed as in Example 2.1. The weights w_s , w_q and $w_\mu \in \mathbb{R}^+$ play the same role as in Example 2.1. W.l.o.g., both the query and the feature vectors can be assumed to be normalized to lie on the unit sphere. We give an example in three dimensions in Figure 1(b), showing locations of tuples extracted from three different relations R_1 , R_2 , and R_3 (in blue, red, and green, respectively). For ease of illustration, the points are obtained from those in Table I by setting azimuth and elevation as $\mathbf{x}(\tau_i)[\theta, \phi] \leftarrow \mathbf{q}[\theta, \phi] + \pi/20 \cdot \mathbf{x}(\tau_i)[\theta, \phi]$, where $\mathbf{q}[\theta, \phi] = [\pi/4, \pi/4]$. The figure also shows three circles on the unit sphere corresponding to the loci of points with the same cosine similarity with the query as the last accessed tuple from each relation.

Definition 2.3. The *proximity rank join* problem is an $(n+2)$ -tuple (R_1, \dots, R_n, S, K) such that S is an aggregation function defined as in (1), all the relations R_1, \dots, R_n are accessed sequentially either in: A) increasing order of $\delta(\cdot, \mathbf{q})$, or B) decreasing order of $\sigma(\cdot)$, and $1 \leq K \leq |R_1 \times \dots \times R_n|$.

A solution is an ordered relation O containing the top K combinations from $R_1 \times \dots \times R_n$ ordered by S (score ties are resolved using a tie-breaking criterion). A proximity rank join algorithm that on input (R_1, \dots, R_n, S, K) returns the corresponding O upon termination is said to be *correct*.

Proximity rank join closely resembles *rank join* [Ilyas et al. 2004]. Unlike the latter, the former assumes that *i*) each tuple is equipped with a feature vector, *ii*) the aggregation function depends both on the score and on the feature vector, and *iii*) the relations are accessed as specified in Definition 2.3.

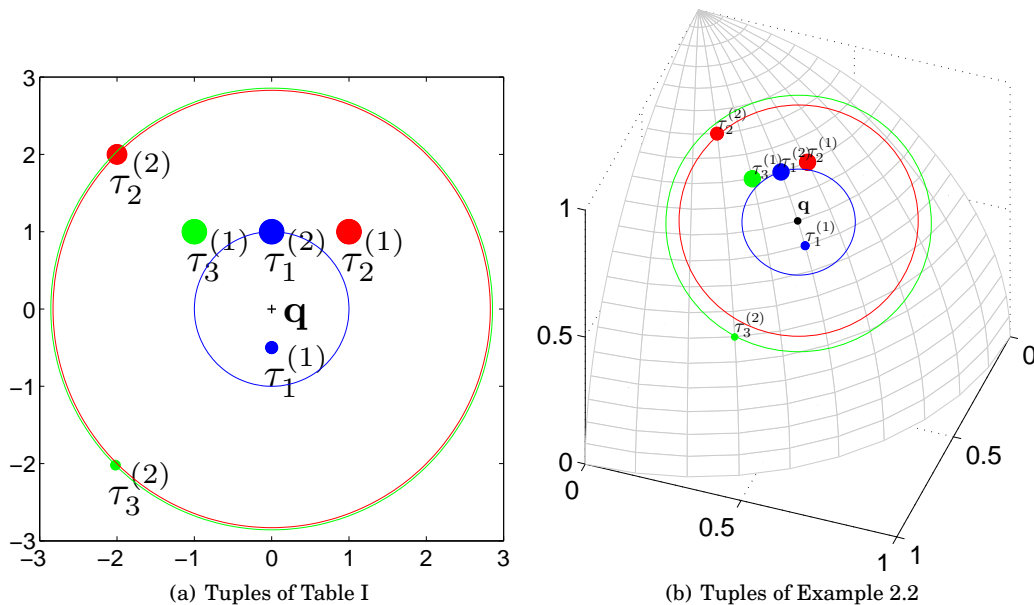


Fig. 1. (a) Location of the tuples of Table I, represented as discs whose radius is proportional to the score. (b) Location of the tuples of Example 2.2.

The proximity rank join problem can be tackled by adopting the *ProxRJ* template reported in Algorithm 1, which adapts the *Pull/Bound Rank Join (PBRJ)* template originally introduced for rank join in [Schnaitter and Polyzotis 2008]. The *chooseInput* function (line 4) of a given *pulling strategy PS* decides at each step the next relation to be accessed. The *updateBound* function (line 9) of a given *bounding scheme BS* computes an upper bound on the aggregate score of the unseen combinations. The unseen combinations are those that can be formed with at least an unseen tuple $\tau_i \in R_i - P_i$.

The above-mentioned differences between rank join and proximity rank join on the assumptions regarding the relations and the aggregation function have an impact on the computation of the upper bound (and, consequently, on the pulling strategy). This will be further discussed in Section 3.

Unlike the general *PBRJ* template, *ProxRJ* focuses on cross-product (line 6), the notion of join being here implicitly captured by the proximity between the objects [Silva et al. 2010]. Yet, the results of this paper also hold when an additional join predicate is used to select a subset of the cross-product.

An algorithm complying with the *ProxRJ* template is correct (as *PBRJ* is for rank join [Schnaitter and Polyzotis 2008]) if: *i)* *updateBound* returns a correct upper bound on the aggregate scores of the combinations that use at least one unread tuple; *ii)* *chooseInput* returns an index of an unexhausted relation.

We define the notion of tight bounding scheme as in [Schnaitter and Polyzotis 2008], but we drop their condition (C3), which is only relevant when tuples may have multiple scores and when the behavior of \mathcal{S} is only assumed to be known on the observed tuples.

Definition 2.4. Let $I = (R_1, \dots, R_n, \mathcal{S}, K)$ be a proximity rank join problem and let P_i be the extracted portion of R_i , $1 \leq i \leq n$:

- A continuation of I is a problem $(R'_1, \dots, R'_n, \mathcal{S}, K)$ that satisfies the following conditions.

ALGORITHM 1: *ProxRJ*($R_1, \dots, R_n, \mathcal{S}, K$)

Input : relations R_1, \dots, R_n ; function S as in (1); result size K
Output: K combinations with highest aggregate score
Data : input buffers P_1, \dots, P_n ; output buffer O

```

1 begin
2    $t \leftarrow \infty$ ;
3   while  $|O| < K$  or  $\min_{\omega \in O} \mathcal{S}(\omega) < t$  do
4      $i \leftarrow PS.chooseInput()$ ;
5      $\tau_i \leftarrow$  next unseen tuple of  $R_i$ ;
6      $R \leftarrow P_1 \times \dots \times P_{i-1} \times \{\tau_i\} \times P_{i+1} \times \dots \times P_n$ ;
7     Add each member of  $R$  to  $O$ , retaining only the top  $K$  combinations;
8     Add  $\tau_i$  to  $P_i$ ;
9      $t \leftarrow BS.updateBound(\tau_i)$ ;
10  end
11  return  $O$ 
12 end

```

(C1) The first $|P_i|$ tuples in R'_i and R_i are identical.

(C2) $|R'_i| = |R_i|$ for all i .

- If τ is a combination for some continuation of I , and τ uses at least one unseen tuple, then we say τ is a *potential result* and $\mathcal{S}(\tau)$ is a *potential score* of I .
- A bounding scheme is tight if *updateBound* returns a potential score or $-\infty$ whenever $|P_1 \times \dots \times P_n| \geq K$.

In order to characterize the performance of proximity rank join algorithms, we introduce the *sumDepths* cost metric and the notion of instance optimality, which are common in top-k query answering. Let $depth(A, I, i)$ be the depth on input relation R_i explored by algorithm A on problem $I = (R_1, \dots, R_n, \mathcal{S}, K)$ before returning a solution. We define $sumDepths(A, I)$ as $\sum_{i=1}^n depth(A, I, i)$, which provides an indication of the amount of I/O performed by A to solve I . Let \mathcal{A} be the class of correct, deterministic proximity rank join algorithms, and let \mathcal{I} be the class of problems satisfying Definition 2.3. We say that A is *instance-optimal* wrt. \mathcal{A} and \mathcal{I} for the *sumDepths* cost metric if there exist constants c_1 and c_2 such that $sumDepths(A, I) \leq c_1 \cdot sumDepths(A', I) + c_2$ for all $A' \in \mathcal{A}$ and $I \in \mathcal{I}$.

3. BOUNDING SCHEMES FOR DISTANCE-BASED ACCESS

In this section, we carry out our analysis for the case of distance-based access. First, we consider a simple bounding scheme that does not leverage the geometrical constraints imposed by the problem formulation; this bounding scheme is equivalent to that of *HRJN* [Ilyas et al. 2004]. Then, we show how to define a tight upper bound for the score of unseen combinations, which is of paramount importance in order to guarantee instance optimality, as proved in [Schnaitter and Polyzotis 2008].

Similar results and algorithms are available for the simpler case of score-based access, as shown in Section 5.

3.1. Corner bound

Under distance-based access, it is possible to compute a correct upper bound t_c by keeping track of the distance from the query \mathbf{q} of the first and last accessed tuple from each relation, i.e., $\delta(\mathbf{x}(R_i[1]), \mathbf{q})$ and $\delta(\mathbf{x}(R_i[p_i]), \mathbf{q})$, respectively:

$$t_c = \max\{t_1, \dots, t_n\}, \text{ with } t_i = f(\bar{\mathcal{S}}_1, \dots, \underline{\mathcal{S}}_i, \dots, \bar{\mathcal{S}}_n) \quad (6)$$

In (6), \bar{S}_j is an upper bound on the proximity weighted score that can be attained by a tuple $\tau_j \in R_j$, i.e.

$$\bar{S}_j = g_j(\sigma_j^{\max}, \delta(\mathbf{x}(R_j[1]), \mathbf{q}), 0) \quad (7)$$

where σ_j^{\max} is the maximum score possible for R_j . Similarly \underline{S}_i denotes an upper bound on the proximity weighted score that can be attained by an unseen tuple $\tau_i \in R_i - P_i$:

$$\underline{S}_i = g_i(\sigma_i^{\max}, \delta(\mathbf{x}(R_i[p_i]), \mathbf{q}), 0) \quad (8)$$

Note that when no object has been extracted from a relation R_i , i.e., $p_i = 0$, both $\delta(\mathbf{x}(R_i[1]), \mathbf{q})$ and $\delta(\mathbf{x}(R_i[p_i]), \mathbf{q})$ are conventionally set to zero.

The bound in (6), corresponding to the one used by *HRJN*, is a *corner bound* [Schnaitter and Polyzotis 2008]. Bound (6) is not tight, since a possible combination whose aggregate score is equal to the bound might not exist. Here, non-tightness precludes instance optimality.

THEOREM 3.1. *Let A be an algorithm complying with the *ProxRJ* template for distance-based access using the corner bound (6). Then A is not instance optimal for problems with at least two relations.*

PROOF. Consider \mathcal{S} as in (3) with $w_s = 0$, $w_q = 1$, $w_\mu = 1$, and $\mathbf{q} = \mathbf{0}$, and a problem $I = (R_1, R_2, \mathcal{S}, 1)$ with

$$\begin{aligned} \mathbf{x}(\tau_1^{(1)}) &= [0, -0.5]^T & \mathbf{x}(\tau_2^{(1)}) &= [0, 2]^T \\ \mathbf{x}(\tau_1^{(2)}) &= [0, 1]^T & \mathbf{x}(\tau_2^{(2)}) &= [-2, 2]^T \end{aligned}$$

These points are shown in Figure 2. (Tuple scores are immaterial to \mathcal{S} as $w_s = 0$.)

When $p_1 = 2$ and $p_2 = 1$, by reasoning on the geometry of the seen tuples, an algorithm may correctly return $\tau = \tau_1^{(2)} \times \tau_2^{(1)}$ as the top combination for I , since no formable combination can have a higher score than $\mathcal{S}(\tau) = -5.5$. The same conclusion can also be drawn by finding a tight bound as shown in Section 4.1.

Conversely, we show that, in order to terminate, A needs to explore R_1 until a depth that is a priori unbounded. Indeed, when $p_1 = p_2 = 2$, A would compute the upper bounds $t_1 = -5$, $t_2 = -8.25$, and thus $t_c = -5$. With this, τ cannot be guaranteed to be the top combination for I , as $t_c > \mathcal{S}(\tau)$. Since deepening on R_2 cannot lower t_c , we discuss how t_c varies as p_1 grows. Indeed, t_c remains above -5.5 until the first tuple $\tau_1^{(j)}$ in R_1 is seen whose distance from \mathbf{q} is at least $\sqrt{5.5 - \|\mathbf{x}(\tau_2^{(1)}) - \mathbf{q}\|^2} = \sqrt{1.5}$. Since there can be an arbitrary number of tuples in R_1 between $\tau_1^{(2)}$ and $\tau_1^{(j)}$ (which would fall on the dashed circumference in Figure 2), the value of $\text{depth}(A, I, 1)$ upon termination of A is unbounded, hence the claim. \square

A similar result was shown in [Schnaitter and Polyzotis 2008] for *PBRJ* algorithms using the corner bound on the rank join problem. However, Theorem 3.1 does not descend from their results. Indeed, the lack of tightness in rank join derives from failure of the join predicate, whereas here we have cross-products, and the geometry of the problem (absent in rank join) contributes to determining the bounds. Moreover, unlike theirs, our result also holds with two input relations ($n = 2$), as the corner bound turns out to be tight only in the extreme case $n = 1$.

3.2. Tight bound

Let $M = \{i_1, \dots, i_m\}$ denote a proper subset of $\{1, \dots, n\}$ and $m = |M|$, $0 \leq m < n$. Let $PC(M) = P_{i_1} \times \dots \times P_{i_m}$ denote the set of partial combinations that can be formed using

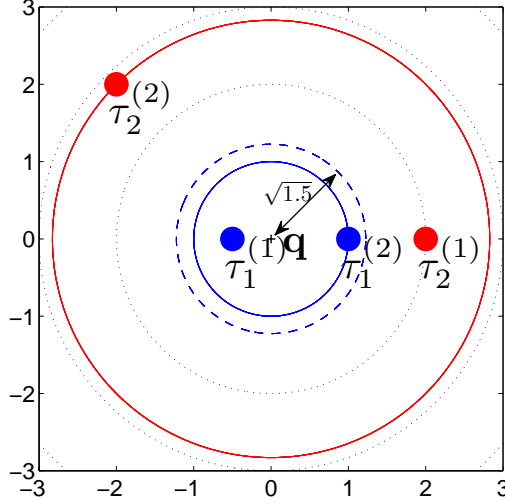


Fig. 2. Geometry of the example used in the proof of Theorem 3.1.

seen tuples from P_i , $i \in M$. Clearly, $|PC(M)| = \prod_{i \in M} p_i$. Given a partial combination $\tau \in PC(M)$, we want to find the maximum aggregate score that can be achieved by completing τ with unseen tuples from $R_i - P_i$, $i \in \{1, \dots, n\} - M$. Distance-based access imposes that $\delta(\mathbf{x}(R_i[r_i]), \mathbf{q}) \geq \delta_i$, for $r_i > p_i$, where $\delta_i = \delta(\mathbf{x}(R_i[p_i]), \mathbf{q})$ is the distance from the query of the last accessed tuple from R_i . The goal is to find the feasible locations $\mathbf{y}_i = \mathbf{x}(\tau_i)$, $i \in \{1, \dots, n\} - M$ of the unseen tuples that maximize the aggregate score, i.e.,

$$\begin{aligned} & \text{maximize} && f(\mathcal{S}_1, \dots, \mathcal{S}_n) \\ & \text{subject to} && \delta(\mathbf{y}_i, \mathbf{q}) \geq \delta_i, i \in \{1, \dots, n\} - M \end{aligned} \quad (9)$$

where

$$\mathcal{S}_i = \begin{cases} g_i(\sigma(\tau_i), \delta(\mathbf{x}(\tau_i), \mathbf{q}), \delta(\mathbf{x}(\tau_i), \boldsymbol{\mu})), & i \in M \\ g_i(\sigma_i^{\max}, \delta(\mathbf{y}_i, \mathbf{q}), \delta(\mathbf{y}_i, \boldsymbol{\mu})), & i \in \{1, \dots, n\} - M \end{cases} \quad (10)$$

Note that all the $n - m$ optimization variables \mathbf{y}_i influence the functions \mathcal{S}_i , $i = 1, \dots, n$, since they participate in the computation of the combination centroid $\boldsymbol{\mu}$.

Let \mathbf{y}_i^* , $i \in \{1, \dots, n\} - M$, denote a solution of (9) for a partial combination $\tau \in PC(M)$, and f^* the value of the objective function computed at \mathbf{y}_i^* . Thus, the upper bound $t(\tau)$ on the aggregate score of the possible combinations formed by completing τ is given by $t(\tau) = f^*$.

Then, for each proper subset M of $\{1, \dots, n\}$, we compute

$$t_M = \max\{t(\tau) | \tau \in PC(M)\} \quad (11)$$

and we obtain the final upper bound as

$$t = \max\{t_M | M \subset \{1, \dots, n\}\} \quad (12)$$

Algorithm 2 provides the pseudocode of the function *updateBound* that is executed for updating the value of the tight bound after each retrieved tuple τ_i . The loop at line 3 evaluates (12). The loop at line 5 evaluates (11) for all partial combinations $\tau' \in PC(M)$. When $M = \emptyset$, the set $PC(M)$ conventionally contains exactly one tuple (the so-called *empty tuple* $\langle \rangle$) and the algorithm may proceed as in all other cases,

since the corresponding combinations are those formed using unseen tuples from all relations. Note that, when invoking *updateBound* after accessing a tuple $\tau_i \in R_i$, these operations need not be performed for all partial combinations. Indeed, the upper bound $t(\tau')$ is computed only if: either τ' is a newly formed partial combination that uses τ_i or τ' is a seen partial combination that can be completed with tuples from R_i (line 6). In all other cases, the value of $t(\tau')$ is not recomputed; instead, we reuse the previously computed value, that we store as a global variable.

ALGORITHM 2: *updateBound*(τ_i) distance-based case

Input : last seen tuple $\tau_i = R_i[p_i]$; seen tuples $P_j, j = 1, \dots, n$, curr. values of $t(\cdot)$ for all seen combinations
Output: Tight upper bound t

```

1 begin
2    $t \leftarrow -\infty$ ;
3   for  $M \subset \{1, \dots, n\}$  do
4      $t_M \leftarrow -\infty$ ;
5     for  $\tau' \in PC(M)$  do
6       if  $(i \in M \wedge \tau'_i = \tau_i) \vee i \notin M$  then
7         Compute  $t(\tau')$  solving (9);
8       end
9        $t_M \leftarrow \max\{t_M, t(\tau')\}$ ;
10    end
11     $t = \max\{t, t_M\}$ ;
12  end
13  return  $t$ 
14 end

```

THEOREM 3.2. *The bounding scheme (12) is tight.*

PROOF. In order to verify Definition 2.4 for any given proximity rank join problem I and extracted portions of the relations, it suffices to exhibit a continuation of I in which a combination using at least one unseen tuple has a score that equals the bound given by (12). To do that, let $\mathbf{y}_i^*, i \in \{1, \dots, n\} - M$, denote a solution of problem (9) for the partial combination $\tau \in PC(M)$ maximizing (11) with the set M maximizing (12). It suffices then to extend each relation $R_i, i \in \{1, \dots, n\} - M$, with a tuple with score σ_i^{\max} and feature vector \mathbf{y}_i^* . With this, *i*) the value of the aggregation function for the combination made by τ and the new tuples equals by construction the bound (12), and *ii*) the requirements posed by distance-based access are met by satisfaction of the constraints in (9). \square

The simplest pulling strategy is the one that accesses the inputs in a *round-robin* fashion (e.g., in the order R_1, \dots, R_n). Tightness and a round-robin strategy are sufficient to guarantee instance optimality.

THEOREM 3.3 (THEOREM 5.1 OF [SCHNAITTER AND POLYZOTIS 2008]). *Let F be an instantiation of PBRJ with the round-robin pulling strategy and a tight bounding scheme. Then F is instance-optimal with optimality ratio n within instances with n input relations.*

This result seamlessly adapts to our case, as stated in Theorem 3.4 below.

THEOREM 3.4. *ProxRJ with the tight bounding scheme (12) and a round-robin pulling strategy is instance-optimal for proximity rank join with distance-based access.*

PROOF. Follows immediately from tightness of (12) and Theorem 3.3, as *ProxRJ* is a specialization of *PBRJ*. \square

We observe that the computation of the tight upper bound (12) comes at a cost that is polynomial under *data complexity* (i.e., w.r.t. the number of retrieved tuples). In fact, it suffices to solve the problem (9) a number of times equal to $C = \sum_{m=0}^{n-1} \binom{n}{m} \prod_{j \in M, M \subset \{1, \dots, n\}, |M|=m} p_j$, and it is easily shown that

$$\sum_{m=0}^{n-1} \binom{n}{m} = 2^n - 1 \leq C \leq p^n \sum_{m=0}^{n-1} \binom{n}{m} = p^n (2^n - 1) \quad (13)$$

where $p = \max\{p_1, \dots, p_n\}$. This is aligned with the findings in [Schnaitter and Polyzotis 2008], where the authors also show polynomial data complexity of tight bounding for rank join. Observe that, here too, the number of relations n weighs exponentially on the cost.

In addition, solving each instance of the problem in (9) might be difficult, depending on the aggregation function and the proximity measure in use. In Section 4, we show that, for some relevant cases, such as when using the aggregation function in (3) or in (4), the problem in (9) can be addressed in an efficient way.

Example 3.5. For the relations in Table I, Table II shows the partial combinations that need to be examined, for each of the subsets M of $\{1, 2, 3\}$. For each partial combination τ , we also report the value of $t(\tau)$, which is obtained by solving (9) with the aggregation function (3) (an effective solution of the problem will be shown in Section 4.1). The values of t_M are computed as in (11). Thus, the upper bound t , using (3), is -7 , which can be potentially achieved by an unseen combination formed by completing the seen partial combination $\tau_2^{(1)} \times \tau_3^{(1)}$ with an unseen tuple from R_1 . Therefore, the seen combination $\tau_1^{(2)} \times \tau_2^{(1)} \times \tau_3^{(1)}$ in Table I can be guaranteed to be the top-1, since its aggregate score -7 is as high as t . Note that none of the seen combinations in Table I can be guaranteed to be the top-1 using the corner bound, since, by (6), we have $t_c = \max\{-5, -10.25, -10.25\} = -5$.

3.2.1. Dominance. The solution of problem (9) for a partial combination $\tau \in PC(M)$ depends on the current distance lower bounds δ_i , $i \in \{1, \dots, n\} - M$, and needs to be re-computed whenever any of these lower bounds may change, i.e., after every access to R_i , $i \in \{1, \dots, n\} - M$. Nevertheless, in order to determine t_M as of (11), it is unnecessary to solve (9) for all partial combinations $\tau \in PC(M)$. Indeed, some of them might be dominated, in the sense that, for a dominated partial combination τ , it will never happen that $t_M = t(\tau)$.

In order to determine whether a partial combination is dominated, we consider the score of the K -th object stored in O . That is, we check the dominance if $|O| = K$, and flag a partial combination as dominated whenever

$$t(\tau) \leq \min_{\omega \in O} \mathcal{S}(\omega) \quad (14)$$

Indeed, as more distance-based accesses are made, the values δ_i , $i = 1, \dots, n$, are non-decreasing. Therefore, the feasible region of problem (9) shrinks and the value of the objective function at the optimum, i.e., $t(\tau)$, decreases. If, at some point during the execution, the upper bound on the score of a partial combination does not exceed the score of the combination with the least score in O , it will never do so.

Algorithm 3 refines Algorithm 2 by adding the dominance check in the computation of the tight bound. Within the loop at line 5, the computation of the upper bound $t(\tau')$ for any partial combination $\tau' \in PC(M)$ is avoided, in addition to the cases already

Table II. Partial combinations formed with the tuples of Table I.

M	$\tau \in PC(M)$	$t(\tau)$	t_M
\emptyset	$\langle \rangle$	-19.2	-19.2
{1}	$\tau_1^{(1)}$	-20.6	-19.2
	$\tau_1^{(2)}$	-19.2	
{2}	$\tau_2^{(1)}$	-12.8	-12.8
	$\tau_2^{(2)}$	-19.4	
{3}	$\tau_3^{(1)}$	-12.8	-12.8
	$\tau_3^{(2)}$	-20.1	
{1, 2}	$\tau_1^{(1)} \times \tau_2^{(1)}$	-16.0	-13.5
	$\tau_1^{(1)} \times \tau_2^{(2)}$	-24.0	
	$\tau_1^{(2)} \times \tau_2^{(1)}$	-13.5	
	$\tau_1^{(2)} \times \tau_2^{(2)}$	-20.4	
{1, 3}	$\tau_1^{(1)} \times \tau_3^{(1)}$	-16.0	-13.5
	$\tau_1^{(1)} \times \tau_3^{(2)}$	-22.0	
	$\tau_1^{(2)} \times \tau_3^{(1)}$	-13.5	
	$\tau_1^{(2)} \times \tau_3^{(2)}$	-26.4	
{2, 3}	$\tau_2^{(1)} \times \tau_3^{(1)}$	-7.0	-7.0
	$\tau_2^{(1)} \times \tau_3^{(2)}$	-21.0	
	$\tau_2^{(2)} \times \tau_3^{(1)}$	-13.1	
	$\tau_2^{(2)} \times \tau_3^{(2)}$	-26.8	

discussed for Algorithm 2, also if τ' has been flagged as dominated (line 6). In such a case, the value of $t(\tau')$ is not recomputed. The dominance test is executed by checking the condition in (14) immediately after computing the upper bound $t(\tau')$, whenever at least K combinations have been formed (line 9).

ALGORITHM 3: *updateBound*(τ_i) distance-based case with dominance check

Input : last seen tuple $\tau_i = R_i[p_i]$; seen tuples $P_j, j = 1, \dots, n$, curr. values of $t(\cdot)$ for all seen combinations
Output: Tight upper bound t

```

1 begin
2    $t \leftarrow -\infty$ ;
3   for  $M \subset \{1, \dots, n\}$  do
4      $t_M \leftarrow -\infty$ ;
5     for  $\tau' \in PC(M)$  do
6       if  $\tau'$  is not dominated then
7         if  $(i \in M \wedge \tau'_i = \tau_i) \vee i \notin M$  then
8           Compute  $t(\tau')$  solving (9);
9           if  $|O| = K \wedge t(\tau') \leq \min_{\omega \in O} \mathcal{S}(\omega)$  then
10            | Flag  $\tau'$  as dominated;
11            end
12            end
13             $t_M \leftarrow \max\{t_M, t(\tau')\}$ ;
14          end
15        end
16         $t = \max\{t, t_M\}$ ;
17      end
18    return  $t$ 
19 end

```

4. TIGHT BOUNDS FOR SPECIAL CASES WITH DISTANCE-BASED ACCESS

4.1. A case with Euclidean distance

In this section, we focus on aggregation functions of the form given in (3). We show that, in this case, the computation of the upper bound can be performed by solving a set of quadratic programs.

Initially, we make the simplifying assumption that the access kind allows retrieving all tuples within a target maximum distance δ from the query \mathbf{q} . Thus, all unseen tuples in $R_i - P_i$ are constrained to be at a distance of at least δ . W.l.o.g., let us assume $M = \{1, \dots, m\}$ and consider a partial combination $\tau \in PC(M)$. The upper bound $t(\tau)$ is obtained by solving the following instance of (9).

$$\begin{aligned} \max. \quad & \sum_{i=1}^m w_s \ln(\sigma(\tau_i)) - w_q \|\mathbf{x}(\tau_i) - \mathbf{q}\|^2 - w_\mu \|\mathbf{x}(\tau_i) - \boldsymbol{\mu}\|^2 + \\ & \sum_{i=m+1}^n w_s \ln(\sigma_i^{\max}) - w_q \|\mathbf{y}_i - \mathbf{q}\|^2 - w_\mu \|\mathbf{y}_i - \boldsymbol{\mu}\|^2 \\ \text{s.t.} \quad & \|\mathbf{y}_i - \mathbf{q}\| \geq \delta, i \in \{m+1, \dots, n\} \end{aligned} \quad (15)$$

where $\boldsymbol{\mu} = \frac{1}{n} (\sum_{i=1}^m \mathbf{x}(\tau_i) + \sum_{i=m+1}^n \mathbf{y}_i)$. Problem (15) is a non-convex QCQP (quadratically constrained quadratic program). Let $\boldsymbol{\nu} = \frac{1}{m} \sum_{i=1}^m \mathbf{x}(\tau_i)$ denote the centroid of the partial combination τ . The solution can be easily found in closed form observing that the problem is symmetric in the optimization variables \mathbf{y}_i . Thus, the optimal solution must satisfy $\mathbf{y}_{m+1}^* = \dots = \mathbf{y}_n^* = \mathbf{y}^*$. We have (see Appendix A.1):

$$\mathbf{y}^* = \begin{cases} \mathbf{q} + (\boldsymbol{\nu} - \mathbf{q}) \frac{mw_\mu}{mw_\mu + nw_q} & \text{if } \|(\boldsymbol{\nu} - \mathbf{q}) \frac{mw_\mu}{mw_\mu + nw_q}\| \geq \delta \\ \mathbf{q} + (\boldsymbol{\nu} - \mathbf{q}) \frac{\delta}{\|\boldsymbol{\nu} - \mathbf{q}\|} & \text{otherwise} \end{cases} \quad (16)$$

Note that \mathbf{y}^* lies on the ray from the query through $\boldsymbol{\nu}$.

In general, however, we need to solve the problem

$$\begin{aligned} \max. \quad & \sum_{i=1}^m w_s \ln(\sigma(\tau_i)) - w_q \|\mathbf{x}(\tau_i) - \mathbf{q}\|^2 - w_\mu \|\mathbf{x}(\tau_i) - \boldsymbol{\mu}\|^2 + \\ & \sum_{i=m+1}^n w_s \ln(\sigma_i^{\max}) - w_q \|\mathbf{y}_i - \mathbf{q}\|^2 - w_\mu \|\mathbf{y}_i - \boldsymbol{\mu}\|^2 \\ \text{s.t.} \quad & \|\mathbf{y}_i - \mathbf{q}\| \geq \delta_i, i \in \{m+1, \dots, n\} \end{aligned} \quad (17)$$

which differs from (15) because the distances δ_i are not constrained to be equal. This may occur, for example, when the access kind enables retrieving a target number of tuples from each relation R_i . Although the problem is no longer symmetric in the optimization variables, the following holds.

THEOREM 4.1. *In the optimal solution of (17) all the \mathbf{y}_i^* 's, $i = m+1, \dots, n$, are collinear and lie on the ray from the query through the centroid of the partial combination.*

PROOF. W.l.o.g., let $\mathbf{q} = \mathbf{0}$. We prove the claim by contradiction. Assume that the optimal solution is such that $\mathbf{y}_{m+1}^*, \dots, \mathbf{y}_n^*$ are not collinear. We show that there is a feasible solution of (17) that attains a larger value of the objective function, thus contradicting the statement that $\mathbf{y}_{m+1}^*, \dots, \mathbf{y}_n^*$ is the optimal solution. The objective function of problem (17) can be rewritten as a function $f(\mathbf{y}_{m+1}, \dots, \mathbf{y}_n)$ given by

$$\begin{aligned} f(\mathbf{y}_{m+1}, \dots, \mathbf{y}_n) &= \sum_{i=1}^m w_s \ln(\sigma(\tau_i)) - w_q \|\mathbf{x}_i\|^2 - w_\mu \|\mathbf{x}_i - \boldsymbol{\mu}\|^2 + \sum_{i=m+1}^n w_s \ln(\sigma_i^{\max}) + \\ & \quad - w_q \|\mathbf{y}_i\|^2 - w_\mu \|\mathbf{y}_i - \boldsymbol{\mu}\|^2 \\ &= \sum_{i=1}^m w_s \ln(\sigma(\tau_i)) - w_q \|\mathbf{x}_i\|^2 + \sum_{i=m+1}^n w_s \ln(\sigma_i^{\max}) + \\ & \quad - w_q \|\mathbf{y}_i\|^2 - w_\mu h(\mathbf{y}_{m+1}, \dots, \mathbf{y}_n) \end{aligned}$$

where we adopt the shorthand notation $\mathbf{x}_i = \mathbf{x}(\tau_i)$ and $h(\mathbf{y}_{m+1}, \dots, \mathbf{y}_n)$ is given by

$$h(\mathbf{y}_{m+1}, \dots, \mathbf{y}_n) = \sum_{i=1}^m \|\mathbf{x}_i - \boldsymbol{\mu}\|^2 + \sum_{i=m+1}^n \|\mathbf{y}_i - \boldsymbol{\mu}\|^2 \quad (18)$$

As shown in Appendix A.2, we can rewrite $h(\mathbf{y}_{m+1}, \dots, \mathbf{y}_n)$ as follows.

$$h(\mathbf{y}_{m+1}, \dots, \mathbf{y}_n) = \sum_{i=1}^m \mathbf{x}_i^T \mathbf{x}_i + \sum_{i=m+1}^n \mathbf{y}_i^T \mathbf{y}_i - \frac{m^2}{n} \boldsymbol{\nu}^T \boldsymbol{\nu} + \frac{1}{n} \left(\sum_{j=m+1}^n \mathbf{y}_j \right)^T \left(\sum_{j=m+1}^n \mathbf{y}_j \right) - 2 \frac{m}{n} \boldsymbol{\nu}^T \left(\sum_{j=m+1}^n \mathbf{y}_j \right) \quad (19)$$

Let $\mathbf{y}_{m+1}^*, \dots, \mathbf{y}_n^*$ denote the optimal solution, consisting of non-collinear points. Let $\mathbf{y}'_{m+1}, \dots, \mathbf{y}'_n$ denote a possible solution defined as follows:

$$\mathbf{y}'_i = \|\mathbf{y}_i^*\| \frac{\boldsymbol{\nu}}{\|\boldsymbol{\nu}\|} \quad \text{for } m+1 \leq i \leq n \quad (20)$$

Thus, \mathbf{y}'_i has the same distance from the origin as \mathbf{y}_i^* (and thus is feasible) but it lies along the direction of $\boldsymbol{\nu}$.

We can observe that

$$\boldsymbol{\nu}^T \left(\sum_{j=m+1}^n \mathbf{y}'_j \right) > \boldsymbol{\nu}^T \left(\sum_{j=m+1}^n \mathbf{y}_j^* \right)$$

since

$$\sum_{j=m+1}^n \boldsymbol{\nu}^T \mathbf{y}'_j > \sum_{j=m+1}^n \boldsymbol{\nu}^T \mathbf{y}_j^*$$

where the strict inequality stems from the observation that the inner products $\boldsymbol{\nu}^T \mathbf{y}_j$ are maximized when \mathbf{y}_j has the same direction as $\boldsymbol{\nu}$. Also

$$\left(\sum_{j=m+1}^n \mathbf{y}'_j \right)^T \left(\sum_{j=m+1}^n \mathbf{y}'_j \right) > \left(\sum_{j=m+1}^n \mathbf{y}_j^* \right)^T \left(\sum_{j=m+1}^n \mathbf{y}_j^* \right)$$

since

$$\left\| \sum_{j=m+1}^n \mathbf{y}'_j \right\| > \left\| \sum_{j=m+1}^n \mathbf{y}_j^* \right\|$$

where the strict inequality is due to the fact that, given two sets of vectors having pairwise the same lengths (i.e., $\|\mathbf{y}'_i\| = \|\mathbf{y}_i^*\|$, for $m+1 \leq i \leq n$), the length of the sum is maximized when the vectors are parallel. Thus we have

$$h(\mathbf{y}_{m+1}^*, \dots, \mathbf{y}_n^*) > h(\mathbf{y}'_{m+1}, \dots, \mathbf{y}'_n) \quad (21)$$

$$f(\mathbf{y}_{m+1}^*, \dots, \mathbf{y}_n^*) < f(\mathbf{y}'_{m+1}, \dots, \mathbf{y}'_n) \quad (22)$$

which contradicts the statement that $\mathbf{y}_{m+1}^*, \dots, \mathbf{y}_n^*$ is the optimal solution, hence the optimal solution is collinear. \square

By Theorem 4.1, we can reduce (17) to a 1-dimensional problem with n scalar variables θ_i , $i = 1, \dots, n$, representing the distance from \mathbf{q} . The first m variables are constrained to be equal to the length of the orthogonal projection of $\mathbf{x}(\tau_i)$ onto the line defined by the query and the centroid of the seen tuples, i.e., $\theta_i = \mathcal{P}(\mathbf{x}(\tau_i))$, $i = 1, \dots, m$, where

$$\mathcal{P}(\mathbf{x}(\tau_i)) = \frac{(\mathbf{x}(\tau_i) - \mathbf{q})^T (\boldsymbol{\nu} - \mathbf{q})}{\|\boldsymbol{\nu} - \mathbf{q}\|} \quad (23)$$

The remaining $n - m$ variables are lower bounded by the current distances from \mathbf{q} , i.e., $\theta_i \geq \delta_i, m + 1, \dots, n$. The upper bound $t(\tau)$ is obtained by solving the following problem

$$\begin{aligned} \max. \quad & \sum_{i=1}^m w_s \ln(\sigma(\tau_i)) + \sum_{i=m+1}^n w_s \ln(\sigma_i^{\max}) - \sum_{i=1}^n w_q \theta_i^2 - \sum_{i=1}^n w_\mu (\theta_i - \frac{1}{n} \sum_{j=1}^n \theta_j)^2 \\ \text{s.t.} \quad & \theta_i = \mathcal{P}(\mathbf{x}(\tau_i)), \quad i = 1, \dots, m \\ & \theta_i \geq \delta_i, \quad i = m + 1, \dots, n \end{aligned} \quad (24)$$

In Appendix A.3 we show that (24) can be written as a convex quadratic program (QP) with linear constraints, thus it can be efficiently solved using off-the-shelf solvers. Let $\theta^* = [\theta_1^*, \dots, \theta_n^*]^T$ denote the optimal solution of (24). The solution of the original problem (17) is given by

$$\mathbf{y}_i^* = \mathbf{q} + \theta_i^* \frac{\boldsymbol{\nu} - \mathbf{q}}{\|\boldsymbol{\nu} - \mathbf{q}\|}, \quad i = m + 1, \dots, n \quad (25)$$

i.e., the i -th variable is at distance θ_i^* from the query \mathbf{q} and on the ray that originates from \mathbf{q} and goes through $\boldsymbol{\nu}$.

Example 4.2. Assume Table I reports all the seen tuples. Thus, $\delta_1 = 1, \delta_2 = 2\sqrt{2}$ and $\delta_3 = 2\sqrt{2}$. Solving (17) for the partial combination $\tau_2^{(1)}$ gives $\mathbf{y}_1^* = [\sqrt{2}/2, \sqrt{2}/2]^T$ and $\mathbf{y}_3^* = [2, 2]^T$ (and $t(\tau_2^{(1)}) = -12.8$), which lie along the ray from \mathbf{q} to $\mathbf{x}(\tau_2^{(1)})$. Solving (17) for $\tau_1^{(1)} \times \tau_3^{(1)}$ requires: *i*) computing the centroid of $\tau_1^{(1)} \times \tau_3^{(1)}$ ($\boldsymbol{\nu} = [-0.5, 0.25]^T$); *ii*) computing the projections on the line from \mathbf{q} to $\boldsymbol{\nu}$ ($\theta_1 = -0.22, \theta_3 = 1.34$); *iii*) solving (24) to obtain $\theta_2^* = 2\sqrt{2}$; *iv*) computing $\mathbf{y}_2^* = [-2.53, 1.26]^T$ according to (25); *v*) computing $t(\tau_1^{(1)} \times \tau_3^{(1)}) = -16$. Figure 3 shows that the optimal locations of the unseen tuples are, in this case, at the minimum allowed distances, but this does not hold in general.

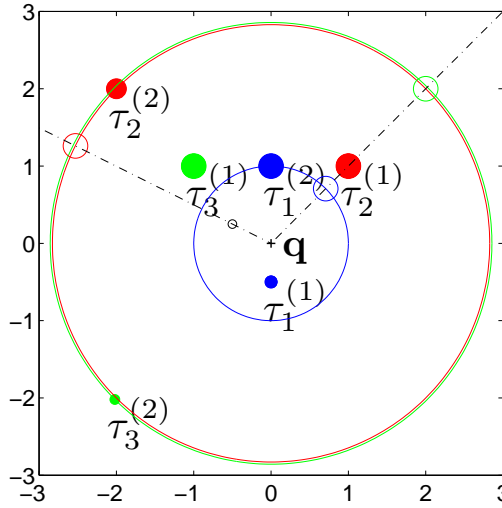


Fig. 3. Solution of problem (17) with the tuples of Table I for: (i) partial combination $\tau_2^{(1)}$; (ii) partial combination $\tau_1^{(1)} \times \tau_3^{(1)}$ whose centroid is indicated by a black empty circle. The optimal locations of unseen tuples are represented by empty circles collinear with the the centroid and the query.

4.2. A case with cosine similarity

In this section, we focus on the computation of a tight bound for an aggregation function based on cosine similarity. Our plan of attack is as follows. First, we formulate an optimization problem to determine an upper bound for each partial combination. Then we show that, in the optimal solution, the vectors to be placed must lie on the plane determined by the origin, the query, and the centroid of the (partial) combination. This allows us to reduce the problem to a 2-dimensional problem, for which an approximate solution can be found by grid search. An alternative approach consists in finding an exact solution for a variant of the problem, stated as a quadratic problem, which provides a correct upper bound very close to the actual tight bound.

We focus now on aggregation functions of the form given in (4), assuming w.l.o.g., as mentioned, that both the query and the feature vectors are normalized to lie on the unit sphere, i.e., $\|\mathbf{q}\| = 1$ and $\|\mathbf{x}(\tau_i)\| = 1$. Note therefore that $\|\boldsymbol{\mu}(\tau)\| \leq 1$, as $\boldsymbol{\mu}(\tau)$ is a convex combination of unit-norm vectors.

W.l.o.g., let us assume $M = \{1, \dots, m\}$ and consider a partial combination $\tau \in PC(M)$. The upper bound $t(\tau)$ is obtained by solving the following instance of (9)

$$\begin{aligned} \max. \quad & \sum_{i=1}^m w_s \sigma(\tau_i) + w_q \mathbf{q}^T \mathbf{x}(\tau_i) + w_\mu \frac{\boldsymbol{\mu}^T \mathbf{x}(\tau_i)}{\|\boldsymbol{\mu}\|} + \sum_{i=m+1}^n w_s \sigma_i^{\max} + w_q \mathbf{q}^T \mathbf{y}_i + w_\mu \frac{\boldsymbol{\mu}^T \mathbf{y}_i}{\|\boldsymbol{\mu}\|} \\ \text{s.t.} \quad & 1 - \mathbf{q}^T \mathbf{y}_i \geq \delta_i, i \in \{m+1, \dots, n\} \\ & \mathbf{y}_i^T \mathbf{y}_i = 1, \end{aligned} \quad (26)$$

where, as usual, σ_i^{\max} indicates the maximum score possible for relation R_i , and, for compactness, the centroid $\boldsymbol{\mu}(\tau)$ is written $\boldsymbol{\mu}$. Similarly to the case with Euclidean distance, it is possible to reduce the size of the problem, thanks to Theorem 4.3 below.

THEOREM 4.3. *In the optimal solution of (26) each \mathbf{y}_i^* , $i = m+1, \dots, n$, is of the form*

$$a_i \mathbf{q} + b_i \boldsymbol{\mu} \quad (27)$$

where $a_i, b_i \in \mathbb{R}$.

PROOF. We prove the claim by contradiction. Assume that the optimal solution is such that not all of $\mathbf{y}_{m+1}^*, \dots, \mathbf{y}_n^*$ are of form (27). We show that there is a feasible solution $\bar{\mathbf{y}}_{m+1}, \dots, \bar{\mathbf{y}}_n$ of (26), with the additional constraints $\mathbf{q}^T \bar{\mathbf{y}}_i = \mathbf{q}^T \mathbf{y}_i^*$, $i \in \{m+1, \dots, n\}$, such that: *i)* $\bar{\mathbf{y}}_i$ is of form (27), $i \in \{m+1, \dots, n\}$ and *ii)* it attains a larger value of the objective function, thus contradicting the statement that $\mathbf{y}_{m+1}^*, \dots, \mathbf{y}_n^*$ is the optimal solution.

The objective function of problem (26) can be rewritten as a function $f(\mathbf{y}_{m+1}, \dots, \mathbf{y}_n)$ defined as

$$\begin{aligned} f(\mathbf{y}_{m+1}, \dots, \mathbf{y}_n) = & \sum_{i=1}^m w_s \sigma(\tau_i) + \sum_{i=m+1}^n w_s \sigma_i^{\max} + \sum_{i=1}^m w_q \mathbf{q}^T \mathbf{x}_i + \\ & \sum_{i=m+1}^n w_q \mathbf{q}^T \mathbf{y}_i + \sum_{i=1}^m w_\mu \frac{\boldsymbol{\mu}^T \mathbf{x}_i}{\|\boldsymbol{\mu}\|} + \sum_{i=m+1}^n w_\mu \frac{\boldsymbol{\mu}^T \mathbf{y}_i}{\|\boldsymbol{\mu}\|}, \end{aligned} \quad (28)$$

where, as usual, we write \mathbf{x}_i instead of $\mathbf{x}(\tau_i)$. Note that the first four terms are constant, given the constraint $\mathbf{q}^T \bar{\mathbf{y}}_i = \mathbf{q}^T \mathbf{y}_i^*$, while the last two terms can be rewritten as follows

$$\frac{w_\mu}{\|\boldsymbol{\mu}\|} \left[\sum_{i=1}^m \boldsymbol{\mu}^T \mathbf{x}_i + \sum_{i=m+1}^n \boldsymbol{\mu}^T \mathbf{y}_i \right] = \frac{w_\mu \boldsymbol{\mu}^T}{\|\boldsymbol{\mu}\|} \left[\sum_{i=1}^m \mathbf{x}_i + \sum_{i=m+1}^n \mathbf{y}_i \right] = \frac{w_\mu \boldsymbol{\mu}^T}{\|\boldsymbol{\mu}\|} [\boldsymbol{\mu}n] = \frac{w_\mu \|\boldsymbol{\mu}\|^2 n}{\|\boldsymbol{\mu}\|} = w_\mu n \|\boldsymbol{\mu}\| \quad (29)$$

Since $\|\boldsymbol{\mu}\|$ is positive and x^2 is monotonically increasing for $x \geq 0$, we can maximize the function in (28) under the given constraints by solving the following equivalent

problem.

$$\begin{aligned} \max. \quad & \|\boldsymbol{\mu}\|^2 \\ \text{s.t.} \quad & \mathbf{q}^T \mathbf{y}_i = \mathbf{q}^T \mathbf{y}_i^*, i \in \{m+1, \dots, n\} \\ & \mathbf{y}_i^T \mathbf{y}_i = 1 \end{aligned} \quad (30)$$

Let $L(\mathbf{y}_{m+1}, \dots, \mathbf{y}_n, \alpha_{m+1}, \dots, \alpha_n, \beta_{m+1}, \dots, \beta_n)$ denote the Lagrangian function associated with (30), that is

$$\begin{aligned} L(\mathbf{y}_{m+1}, \dots, \mathbf{y}_n, \alpha_{m+1}, \dots, \alpha_n, \beta_{m+1}, \dots, \beta_n) = & -\|\boldsymbol{\mu}\|^2 \\ & + \sum_{i=m+1}^n \alpha_i (\mathbf{q}^T \mathbf{y}_i - \mathbf{q}^T \mathbf{y}_i^*) + \sum_{i=m+1}^n \beta_i (\mathbf{y}_i^T \mathbf{y}_i - 1) \end{aligned} \quad (31)$$

where α_i and β_i are the Lagrange multipliers associated with the equality constraints. The Karush-Kuhn-Tucker equations give necessary conditions that need to be satisfied by the solution of (30). More specifically, the stationarity constraints impose that

$$\left. \frac{\partial L(\cdot)}{\partial \mathbf{y}_i} \right|_{\mathbf{y}_i = \bar{\mathbf{y}}_i} = -\frac{2}{n} \boldsymbol{\mu} + \alpha_i \mathbf{q} + 2\beta_i \bar{\mathbf{y}}_i = \mathbf{0} \quad (32)$$

where $\bar{\mathbf{y}}_{m+1}, \dots, \bar{\mathbf{y}}_n$ denotes the optimal solution of (30). Therefore, we have

$$\bar{\mathbf{y}}_i = -\frac{\alpha_i}{2\beta_i} \mathbf{q} + \frac{1}{n\beta_i} \boldsymbol{\mu} \quad (33)$$

This indicates that $\bar{\mathbf{y}}_i$ is of form (27), $i \in \{m+1, \dots, n\}$. Since we assumed that not all of $\mathbf{y}_{m+1}^*, \dots, \mathbf{y}_n^*$ are of form (27), and $\mathbf{y}_{m+1}^*, \dots, \mathbf{y}_n^*$ is also a solution of (30), then $f(\bar{\mathbf{y}}_{m+1}, \dots, \bar{\mathbf{y}}_n) > f(\mathbf{y}_{m+1}^*, \dots, \mathbf{y}_n^*)$. Now, $\bar{\mathbf{y}}_{m+1}, \dots, \bar{\mathbf{y}}_n$ is also a solution for problem (26), hence, $\mathbf{y}_{m+1}^*, \dots, \mathbf{y}_n^*$ cannot be the optimal solution. Contradiction. \square

When \mathbf{q} and $\boldsymbol{\mu}$ are not parallel, the locus π of points of form (27) is a 2-dimensional subspace (a plane) defined by the span of the vectors \mathbf{q} and $\boldsymbol{\mu}$. Theorem 4.3 claims then that all the points of the optimal solution of (26) lie on π . Note that, \mathbf{q} and $\boldsymbol{\mu}$ cannot be parallel if $\delta_i > 0$ for at least one $i \in \{m+1, \dots, n\}$ in problem (26), i.e., if there is at least one constraint on the cosine similarity with the query. Indeed, \mathbf{q} and $\boldsymbol{\mu}$ would be parallel only if the points of the solution were not all on the same side with respect to \mathbf{q} . But, in such a case, an equal or higher (if $w_\mu > 0$) value of the objective function would be obtained by placing all the points on the same side with respect to \mathbf{q} , moving all points from the other side in the symmetric position with respect to \mathbf{q} . In the degenerate case when $\delta_i = 0$ for all $i \in \{m+1, \dots, n\}$, each \mathbf{y}_i^* trivially coincides with \mathbf{q} .

When we compute the tight bound associated with a partial combination τ , we do not yet know the centroid $\boldsymbol{\mu}$, but only the partial centroid $\boldsymbol{\nu}$ (which is defined if M is nonempty). The origin, \mathbf{q} , and $\boldsymbol{\nu}$ also identify a plane (henceforth, the $(\mathbf{q}, \boldsymbol{\nu})$ -plane), if $M \neq \emptyset$, and such a plane turns out to be the same as π , as stated next.

THEOREM 4.4. *Let $\tau \in PC(M)$ with $M \neq \emptyset$. Then, in the optimal solution of (26), each \mathbf{y}_i^* , $i = m+1, \dots, n$, lies on the $(\mathbf{q}, \boldsymbol{\nu})$ -plane.*

PROOF. From Theorem 4.3, we know that each \mathbf{y}_i^* , $i = m+1, \dots, n$, lies on the plane defined by \mathbf{q} , $\boldsymbol{\mu}$, and the origin. Let π be such a plane. To prove the theorem, it suffices to show that π coincides with the $(\mathbf{q}, \boldsymbol{\nu})$ -plane, i.e., that $\boldsymbol{\nu}$ lies on π .

Consider the vector $\boldsymbol{\theta} = \sum_{i=m+1}^n \mathbf{y}_i^*$. All the terms in the sum lie on π , so $\boldsymbol{\theta}$ also lies on π . By definition, the partial centroid can be written as $\boldsymbol{\nu} = \frac{1}{m}(n\boldsymbol{\mu} - \boldsymbol{\theta})$, i.e., $\boldsymbol{\nu}$ is a linear combination of $\boldsymbol{\mu}$ and $\boldsymbol{\theta}$, and thus $\boldsymbol{\nu}$ must also lie on π . \square

In addition to the tuples of Figure 1(b), Figure 4 shows:

- the centroid of the partial combination $\tau_3^{(1)} \times \tau_1^{(1)}$ as a small black circle and the optimal location of an unseen tuple from R_2 (in red) to complete the partial combination;
- the optimal locations of unseen tuples from R_1 (in blue) and R_3 (in green) to complete the partial combination $\tau_2^{(1)}$.

The Figure also shows, as black lines, the intersection of the unit sphere with the (\mathbf{q}, ν) -planes corresponding to the two partial combinations.

Figure 5 shows the projections of the points in Figure 4 onto the two (\mathbf{q}, ν) -planes. In Figure 5(a), the (projected) partial centroid $\tilde{\nu}$ coincides with (the projection of) the only point $(\tilde{\mathbf{x}})$ composing the partial combination $\tau_2^{(1)}$; $\tilde{\mathbf{y}}_1^*$ and $\tilde{\mathbf{y}}_3^*$ are the projections of the optimal locations of the unseen tuples completing the partial combination, and $\tilde{\boldsymbol{\mu}}$ is the projection of the centroid. In Figure 5(b), we indicate similar data for the projection of partial combination $\tau_3^{(1)} \times \tau_1^{(1)}$, given by points $\tilde{\mathbf{x}}_1$ and $\tilde{\mathbf{x}}_3$; here, $\tilde{\mathbf{y}}_2^*$ is the projection of the optimal location of the unseen tuple completing the partial combination.

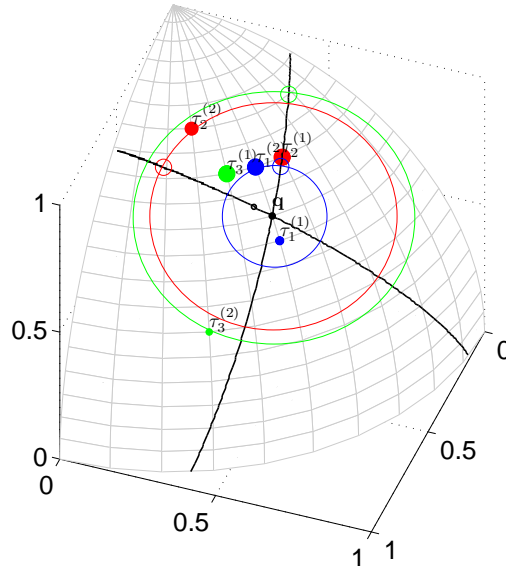
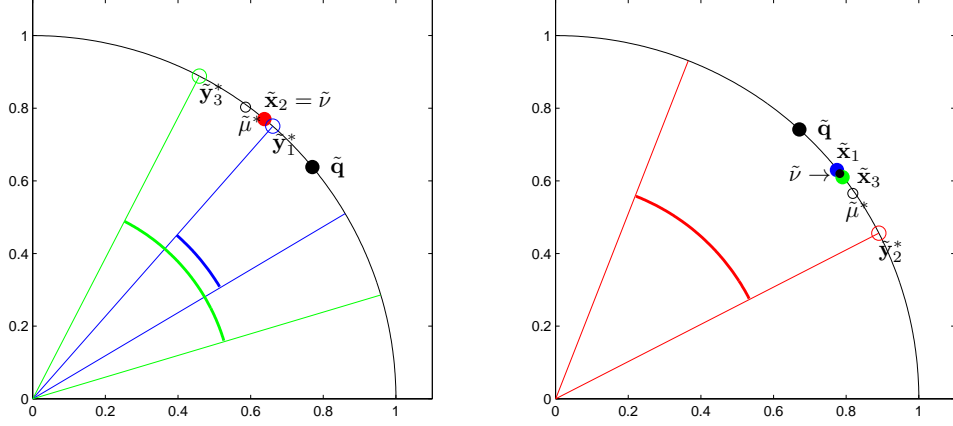


Fig. 4. Solutions of problem (26) with the tuples of Figure 1(b) for partial combination $\tau_2^{(1)}$ and $\tau_3^{(1)} \times \tau_1^{(1)}$ (whose centroid is indicated by a black empty circle). The optimal locations of the missing tuples are represented by empty circles lying on the corresponding (\mathbf{q}, ν) -plane, whose intersection with the unit sphere is indicated with a black line.

Theorem 4.4 suggests to reduce (26) to a 2-dimensional problem with n variables $\tilde{\mathbf{y}}_i \in \mathbb{R}^2$ representing points on the unit circle. Let $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2]$, $\mathbf{u}_i \in \mathbb{R}^d$ denote a basis for the subspace π . If M is nonempty and \mathbf{q} and ν are not parallel, the (\mathbf{q}, ν) -plane is univocally identified and a basis can be found, e.g., by selecting the first two output singular vectors of the singular value decomposition of the $d \times 2$ matrix $[\mathbf{q}, \nu]$. Then, we set $\tilde{\mathbf{x}}_i = \mathbf{U}^T \mathbf{x}_i$, $i = 1, \dots, m$, i.e., equal to the projection of \mathbf{x}_i onto π . When $\tau \in PC(\emptyset)$ or ν is parallel to \mathbf{q} , the (\mathbf{q}, ν) -plane is not univocally identified. In such a case, we simply pick one such plane by setting $\mathbf{u}_1 = \mathbf{q}$ and \mathbf{u}_2 equal to a vector orthogonal to \mathbf{u}_1 . In all



(a) Projection of Figure 4 on the $(\mathbf{q}, \boldsymbol{\nu})$ -plane for partial combination $\tau_2^{(1)}$ (b) Projection of Figure 4 on the $(\mathbf{q}, \boldsymbol{\nu})$ -plane for partial combination $\tau_3^{(1)} \times \tau_1^{(1)}$

Fig. 5. Projections of Figure 4 on the $(\mathbf{q}, \boldsymbol{\nu})$ -planes for the partial combinations.

cases, we define $\tilde{\mathbf{q}} = \mathbf{U}^T \mathbf{q}$, $\tilde{\boldsymbol{\mu}} = \mathbf{U}^T \boldsymbol{\mu}$, and $\tilde{\boldsymbol{\nu}} = \mathbf{U}^T \boldsymbol{\nu}$. The upper bound $t(\boldsymbol{\tau})$ is obtained by solving the following problem

$$\begin{aligned}
 \max. \quad & w_q \sum_{i=1}^n \tilde{\mathbf{q}}^T \tilde{\mathbf{y}}_i + \frac{w_{\boldsymbol{\mu}}}{\|\tilde{\boldsymbol{\mu}}\|} \sum_{i=1}^n \tilde{\boldsymbol{\mu}}^T \tilde{\mathbf{y}}_i + c \\
 \text{s.t.} \quad & \tilde{\mathbf{y}}_i = \tilde{\mathbf{x}}_i, i \in \{1, \dots, m\} \\
 & 1 - \tilde{\mathbf{q}}^T \tilde{\mathbf{y}}_i \geq \delta_i, i \in \{m+1, \dots, n\} \\
 & \tilde{\mathbf{y}}_i^T \tilde{\mathbf{y}}_i = 1, i \in \{1, \dots, n\}
 \end{aligned} \tag{34}$$

where $c = \sum_{i=1}^m w_s \sigma(\tau_i) + \sum_{i=m+1}^n w_s \sigma_i^{\max}$. Note that the vectors $\tilde{\mathbf{y}}_i$, $i \in \{m+1, \dots, n\}$, can be parameterized based on the value of an angle, i.e., $\tilde{\mathbf{y}}_i = [\cos \theta_i, \sin \theta_i]^T$. Therefore, the problem has $n - m \leq n$ scalar unknown variables θ_i .

An approximate solution of (34) can be found by means of a grid search on a $(n - m)$ -dimensional space, i.e., considering P discrete values for each of the variables θ_i , $i \in \{m+1, \dots, n\}$. In this case, the objective function in (34) needs to be evaluated up to P^{n-m} times. Increasing P increases the accuracy of the solution at the cost of additional computation.

An alternative approach consists of computing a correct upper bound, which can be obtained exactly although it might not be tight. To this end, consider a feasible solution $\tilde{\mathbf{y}}_{m+1}, \dots, \tilde{\mathbf{y}}_n$ constructed according to the following criteria: *i*) all the constraints in (34) are satisfied with the equality sign, i.e., $1 - \tilde{\mathbf{q}}^T \tilde{\mathbf{y}}_i = \delta_i$, $\tilde{\mathbf{y}}_i^T \tilde{\mathbf{y}}_i = 1$, $i \in \{m+1, \dots, n\}$. Note that, for each variable, there are two such unit norm vectors $\tilde{\mathbf{y}}_i$, symmetric with respect to the vector $\tilde{\mathbf{q}}$; *ii*) among all the possible 2^{n-m} feasible solutions satisfying the constraints above, select one of the two feasible solutions for which all unit norm vectors $\tilde{\mathbf{y}}_i$ are on the same side as $\tilde{\boldsymbol{\nu}}$ with respect to $\tilde{\mathbf{q}}$. Let $\tilde{\boldsymbol{\mu}} = \frac{m}{n} \boldsymbol{\nu} + \frac{1}{n-m} (\tilde{\mathbf{y}}_{m+1} + \dots + \tilde{\mathbf{y}}_n)$ denote the centroid corresponding to such a feasible solution and consider the following

problem, obtained from (34) by replacing $\|\tilde{\mu}\|$ with the constant value $\|\bar{\mu}\|$.

$$\begin{aligned} \max. \quad & w_q \sum_{i=1}^n \tilde{\mathbf{q}}^T \tilde{\mathbf{y}}_i + \frac{w_\mu}{\|\tilde{\mu}\|} \sum_{i=1}^n \tilde{\mu}^T \tilde{\mathbf{y}}_i + c \\ \text{s.t.} \quad & \tilde{\mathbf{y}}_i = \tilde{\mathbf{x}}_i, i \in \{1, \dots, m\} \\ & 1 - \tilde{\mathbf{q}}^T \tilde{\mathbf{y}}_i \geq \delta_i, i \in \{m+1, \dots, n\} \\ & \tilde{\mathbf{y}}_i^T \tilde{\mathbf{y}}_i = 1, i \in \{1, \dots, n\} \end{aligned} \quad (35)$$

Let f^* and \bar{f}^* denote, respectively, the value attained by the optimal solution of (34) and (35).

It is possible to prove that the following theorem holds

THEOREM 4.5. *\bar{f}^* is a correct upper bound on f^* .*

PROOF. Let $\tilde{\mathbf{y}}_{m+1}^*, \dots, \tilde{\mathbf{y}}_n^*$ and $\bar{\mathbf{y}}_{m+1}^*, \dots, \bar{\mathbf{y}}_n^*$ denote, respectively, the optimal solution of (34) and (35). Let $h(\tilde{\mathbf{y}}_{m+1}, \dots, \tilde{\mathbf{y}}_n) = w_q \sum_{i=1}^n \tilde{\mathbf{q}}^T \tilde{\mathbf{y}}_i + c$, $n(\tilde{\mathbf{y}}_{m+1}, \dots, \tilde{\mathbf{y}}_n) = w_\mu \sum_{i=1}^n \tilde{\mu}^T \tilde{\mathbf{y}}_i$ and $d(\tilde{\mathbf{y}}_{m+1}, \dots, \tilde{\mathbf{y}}_n) = \|\tilde{\mu}\|$.

We can leverage the following lemma, which gives a lower bound on the centroid corresponding to the optimal solution of (34), $\|\tilde{\mu}^*\|$, where $\tilde{\mu}^* = \frac{m}{n} \nu + \frac{1}{n-m} (\tilde{\mathbf{y}}_{m+1}^* + \dots + \tilde{\mathbf{y}}_n^*)$.

LEMMA 4.6. *Let $\bar{\mu} = \frac{m}{n} \nu + \frac{1}{n-m} (\bar{\mathbf{y}}_{m+1} + \dots + \bar{\mathbf{y}}_n)$. Then $\|\tilde{\mu}^*\| \geq \|\bar{\mu}\|$.*

It is possible to find the following relationship between the optimal solutions of (35) and (34).

$$\bar{f}^* = h(\bar{\mathbf{y}}_{m+1}^*, \dots, \bar{\mathbf{y}}_n^*) + \frac{n(\bar{\mathbf{y}}_{m+1}^*, \dots, \bar{\mathbf{y}}_n^*)}{\|\bar{\mu}\|} \geq \quad (36)$$

$$h(\tilde{\mathbf{y}}_{m+1}^*, \dots, \tilde{\mathbf{y}}_n^*) + \frac{n(\tilde{\mathbf{y}}_{m+1}^*, \dots, \tilde{\mathbf{y}}_n^*)}{\|\tilde{\mu}\|} \geq \quad (37)$$

$$h(\tilde{\mathbf{y}}_{m+1}^*, \dots, \tilde{\mathbf{y}}_n^*) + \frac{n(\tilde{\mathbf{y}}_{m+1}^*, \dots, \tilde{\mathbf{y}}_n^*)}{d(\tilde{\mathbf{y}}_{m+1}^*, \dots, \tilde{\mathbf{y}}_n^*)} = f^* \quad (38)$$

where the first inequality is due to the fact that $\bar{\mathbf{y}}_{m+1}^*, \dots, \bar{\mathbf{y}}_n^*$ is the optimal solution of (35), and the second inequality is due to Lemma 4.6. \square

PROOF OF LEMMA 4.6. Consider a feasible solution $\bar{\mathbf{y}}_{m+1}, \dots, \bar{\mathbf{y}}_n$ of problem 34 constructed by satisfying the constraints with the equality, as previously described. Let $\bar{\mu} = \frac{m}{n} \nu + \frac{1}{n} (\bar{\mathbf{y}}_{m+1} + \dots + \bar{\mathbf{y}}_n)$ denote the centroid corresponding to such a feasible solution.

Let $f(\tilde{\mathbf{y}}_{m+1}, \dots, \tilde{\mathbf{y}}_n) = h_q(\tilde{\mathbf{y}}_{m+1}, \dots, \tilde{\mathbf{y}}_n) + h_\mu(\tilde{\mathbf{y}}_{m+1}, \dots, \tilde{\mathbf{y}}_n)$, where $h_q(\tilde{\mathbf{y}}_{m+1}, \dots, \tilde{\mathbf{y}}_n) = w_q \sum_{i=1}^n \tilde{\mathbf{q}}^T \tilde{\mathbf{y}}_i + c$ and $h_\mu(\tilde{\mathbf{y}}_{m+1}, \dots, \tilde{\mathbf{y}}_n) = w_\mu \|\tilde{\mu}^T\|$.

Let $\tilde{\mathbf{y}}_{m+1}^*, \dots, \tilde{\mathbf{y}}_n^*$ denote the optimal solution of (34). Hence

$$f(\tilde{\mathbf{y}}_{m+1}^*, \dots, \tilde{\mathbf{y}}_n^*) \geq f(\bar{\mathbf{y}}_{m+1}, \dots, \bar{\mathbf{y}}_n) \quad (39)$$

The feasible solution $\bar{\mathbf{y}}_{m+1}, \dots, \bar{\mathbf{y}}_n$ is the closest to the query vector $\tilde{\mathbf{q}}$. Hence

$$h_q(\bar{\mathbf{y}}_{m+1}, \dots, \bar{\mathbf{y}}_n) \geq h_q(\tilde{\mathbf{y}}_{m+1}^*, \dots, \tilde{\mathbf{y}}_n^*) \quad (40)$$

From (39) and (40) it follows that

$$h_\mu(\tilde{\mathbf{y}}_{m+1}^*, \dots, \tilde{\mathbf{y}}_n^*) \geq h_\mu(\bar{\mathbf{y}}_{m+1}, \dots, \bar{\mathbf{y}}_n) \quad (41)$$

\square

Note that, if $\bar{\mathbf{y}}_i^* = \bar{\mathbf{y}}_i$, then $\tilde{\mathbf{y}}_i^* = \bar{\mathbf{y}}_i^*$ and $\bar{f}^* = f^*$, i.e., the upper bound computed solving (35) is tight.

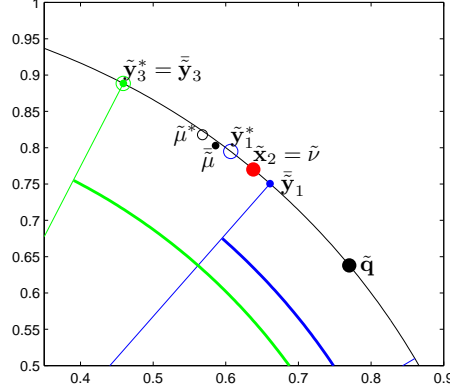


Fig. 6. Variant of Figure 5(a) with different weights in the aggregation function. The optimal solution \tilde{y}_1^* is not placed in the allowed position closest to the query, i.e., \tilde{y}_1 .

Figure 6 shows a detailed view of a variant of the case of Figure 5(a), in which a larger value for weight w_μ than for w_q is used. This causes the optimal solution \tilde{y}_1^* not to lie in the allowed position closest to the query, i.e., \tilde{y}_1 . As can be seen, the centroid $\tilde{\mu}^*$ of the combination found with the optimal solution has a norm that is very similar to the norm of vector $\tilde{\mu}$, with $\|\tilde{\mu}^*\| \geq \|\tilde{\mu}\|$, as proved in Lemma 4.6.

Unlike (34), (35) is a quadratic problem, whose exact solution can be found as illustrated in Appendix A.4.

5. BOUNDING SCHEMES FOR SCORE-BASED ACCESS

Under score-based access, it is possible to compute a correct upper bound by keeping track of the scores $\sigma(R_i[1])$ and $\sigma(R_i[p_i])$ of the first and, respectively, last accessed tuples from each relation. The upper bound is given by

$$t_c^s = \max\{t_1^s, \dots, t_n^s\}, \text{ with } t_i^s = f(\bar{S}_1^s, \dots, \underline{S}_i^s, \dots, \bar{S}_n^s) \quad (42)$$

where \bar{S}_j^s is an upper bound on the proximity weighted score that can be attained by a tuple $\tau_j \in R_j$, i.e.,

$$\bar{S}_j^s = g_j(\sigma(R_i[1]), 0, 0) \quad (43)$$

Similarly \underline{S}_i^s denotes an upper bound on the proximity weighted score that can be attained by an unseen tuple $\tau_i \in R_i - P_i$:

$$\underline{S}_i^s = g_i(\sigma(R_i[p_i]), 0, 0) \quad (44)$$

Such a bound is a corner bound (like that of *HRJN* [Ilyas et al. 2004]), and it does not consider any geometric constraints of the problem at hand. As for the case of distance-based access, the bound is not tight and precludes instance optimality.

THEOREM 5.1. *Let A be an algorithm complying with the *ProxRJ* template for score-based access using the corner bound (42). Then A is not instance optimal for problems with at least two relations.*

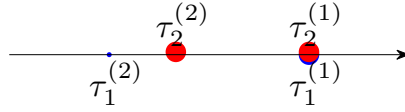


Fig. 7. Geometry of the example used in the proof of Theorem 5.1.

PROOF. Consider an aggregation function of form (3) on a one-dimensional space, with $w_s = w_q = w_\mu = 1$, $\mathbf{q} = \mathbf{0}$, and a problem $I = (R_1, R_2, \mathcal{S}, 1)$ with

$$\begin{aligned} \sigma(\tau_1^{(1)}) &= 1 & \mathbf{x}(\tau_1^{(1)}) &= [1] & \sigma(\tau_2^{(1)}) &= 1 & \mathbf{x}(\tau_2^{(1)}) &= [1] \\ \sigma(\tau_1^{(2)}) &= e^{-5} & \mathbf{x}(\tau_1^{(2)}) &= [0] & \sigma(\tau_2^{(2)}) &= 1 & \mathbf{x}(\tau_2^{(2)}) &= [1/3] \end{aligned}$$

as shown in Figure 7. The combination $\tau = \tau_1^{(1)} \times \tau_2^{(2)}$ has score $\mathcal{S}(\tau) = -4/3$, which is the highest possible for all seen and unseen combinations. However, when $p_1 = p_2 = 2$ the corner bound is $t_c^s = 0$, thus τ cannot be guaranteed to be top by t_c^s , as $t_c^s > \mathcal{S}(\tau)$. Deepening on R_1 cannot lower t_c^s . When p_2 grows, t_c^s remains above $\mathcal{S}(\tau)$ until the first tuple $\tau_2^{(j)}$ is seen with $\sigma(\tau_2^{(j)}) \leq e^{-4/3}$. The number of tuples between $\tau_2^{(2)}$ and $\tau_2^{(j)}$ in R_2 is arbitrary, hence the claim. \square

Similarly to Section 3, let $\tau \in PC(M)$ denote a partial combination that can be formed using seen tuples from R_i , $i \in M$. A tight upper bound $t^s(\tau)$ is obtained by solving the following unconstrained optimization problem in the variables $\mathbf{y}_i \in \mathbb{R}^d$, $i \in \{1, \dots, n\} - M$

$$\begin{aligned} & \text{maximize } f(\mathcal{S}_1^s, \dots, \mathcal{S}_n^s), \text{ where} \\ \mathcal{S}_i^s &= \begin{cases} g_i(\sigma(\tau_i), \delta(\mathbf{x}(\tau_i), \mathbf{q}), \delta(\mathbf{x}(\tau_i), \boldsymbol{\mu})), & i \in M \\ g_i(\sigma(R_i[p_i]), \delta(\mathbf{y}_i, \mathbf{q}), \delta(\mathbf{y}_i, \boldsymbol{\mu})), & i \in \{1, \dots, n\} - M \end{cases} \end{aligned} \quad (45)$$

A tight bound for the score-based case is then given by

$$t^s = \max\{t_M^s | M \subset \{1, \dots, n\}\}, t_M^s = \max\{t^s(\tau) | \tau \in PC(M)\} \quad (46)$$

This suffices to obtain instance optimality. The proofs of these claims are analogous to those of Theorems 3.2 and 3.4.

THEOREM 5.2. *The bounding scheme (46) is tight.*

THEOREM 5.3. *ProxRJ with the tight bounding scheme (46) and a round-robin pulling strategy is instance-optimal for proximity rank join with score-based access.*

Unlike the case of distance-based access, for each set M , we need to keep track of only one partial combination τ_{best}^M that dominates all the others. In other words, if, at a given state of execution, $t^s(\tau^\alpha) \geq t^s(\tau^\beta)$, for two given partial combinations τ^α and τ^β , then the same inequality holds when additional tuples are retrieved from R_i , $i \in \{1, \dots, n\} - M$. Retrieving additional tuples from R_i , $i \in M$ creates new partial combinations. Updating the tight bound when a new tuple is retrieved can be done

incrementally, as it suffices to solve (45) for each new partial combination and to keep track of the partial combination with the highest upper bound.

Algorithm 4 provides the pseudocode that is executed for updating the value of the tight bound after each retrieved tuple τ_i for the case of score-based access. The algorithm includes a dominance test and proceeds as Algorithm 3, with two main differences: *i*) the upper bound of a partial combination is computed according to (45), and *ii*) partial combinations that do not exceed the current value of t_M^s can be immediately flagged as dominated, and will remain so. Note that, for each $M \subset \{1, \dots, n\}$, we keep track of the partial combination τ_{best}^M with the currently highest upper bound, each stored as a global variable. At line 10, if the upper bound of the current partial combination τ' exceeds t_M^s , we flag the previous τ_{best}^M (if defined) as dominated (line 11), we update t_M^s (line 12), and we set τ' as the new τ_{best}^M (line 13).

ALGORITHM 4: *updateBound*(τ_i) score-based case with dominance check

Input : last seen tuple $\tau_i = R_i[p_i]$; seen tuples $P_j, j = 1, \dots, n$, curr. values of $t^s(\cdot)$ for all seen combinations, τ_{best}^M for every $M \subset \{1, \dots, n\}$

Output: Tight upper bound t^s

```

1 begin
2    $t^s \leftarrow -\infty$ ;
3   for  $M \subset \{1, \dots, n\}$  do
4      $t_M^s \leftarrow -\infty$ ;
5     for  $\tau' \in PC(M)$  do
6       if  $\tau'$  is not dominated then
7         if  $i \in M \wedge \tau'_i = \tau_i \vee i \notin M$  then
8           Compute  $t^s(\tau')$  solving (45);
9         end
10        if  $t^s(\tau') > t_M^s$  then
11          Flag  $\tau_{best}^M$  as dominated;
12           $t_M^s \leftarrow t^s(\tau')$ ;
13           $\tau_{best}^M \leftarrow \tau'$ ;
14        else
15          Flag  $\tau'$  as dominated;
16        end
17      end
18    end
19  end
20  return  $t^s$ 
21 end

```

5.1. Tight bound for a case with Euclidean distance

Consider an aggregation function of the form given in (3). W.l.o.g. let us assume $M = \{1, \dots, m\}$ and a partial combination $\tau \in PC(M)$. The (partial) upper bound is obtained by solving the following problem

$$\max. \sum_{i=1}^m w_s \sigma(\tau_i) - w_q \|\mathbf{x}(\tau_i) - \mathbf{q}\|^2 - w_\mu \|\mathbf{x}(\tau_i) - \boldsymbol{\mu}\|^2 + \sum_{i=m+1}^n w_s \sigma(R[p_i]) - w_q \|\mathbf{y}_i - \mathbf{q}\|^2 - w_\mu \|\mathbf{y}_i - \boldsymbol{\mu}\|^2$$

The problem is of the same kind as (15), but without constraints. Therefore, the optimal solution is given by

$$\mathbf{y}_{m+1}^* = \dots = \mathbf{y}_n^* = \mathbf{q} + (\boldsymbol{\nu} - \mathbf{q}) \frac{mw_\mu}{mw_\mu + nw_q} \quad (47)$$

i.e., the upper bound on the score of unseen combinations is achieved by completing the partial combination τ with tuples from R_i , $i = m + 1, \dots, n$ such that: *i*) the unseen tuples are located along the ray that originates from the query \mathbf{q} and goes through the centroid $\boldsymbol{\nu}$ of the partial combination; *ii*) the score is equal to the score of the last seen tuple of R_i .

5.2. Bound for a case with cosine similarity

Consider an aggregation function of the form given in (4). W.l.o.g. let us assume $M = \{1, \dots, m\}$ and a partial combination $\tau \in PC(M)$. The (partial) upper bound is obtained by solving the following problem

$$\begin{aligned} \max. \quad & \sum_{i=1}^m w_s \sigma(\tau_i) + w_q \mathbf{q}^T \mathbf{x}(\tau_i) + w_\mu \frac{\boldsymbol{\mu}^T \mathbf{x}(\tau_i)}{\|\boldsymbol{\mu}\|} + \sum_{i=m+1}^n w_s \sigma(R[p_i]) + w_q \mathbf{q}^T \mathbf{y}_i + w_\mu \frac{\mathbf{y}_i^T \boldsymbol{\mu}}{\|\boldsymbol{\mu}\|} \\ \text{s.t.} \quad & \mathbf{y}_i^T \mathbf{y}_i = 1 \end{aligned} \quad (48)$$

The problem is of the same kind as (26), but replacing σ_i^{\max} with $\sigma(R[p_i])$, and removing the constraints $1 - \mathbf{q}^T \mathbf{y}_i \geq \delta_i$ due to distance-based access (this is equivalent to setting $\delta_i = 0$). It can be observed that the problem is symmetric in the optimization variables \mathbf{y}_i . Thus, the optimal solution must satisfy $\mathbf{y}_{m+1}^* = \dots = \mathbf{y}_n^* = \mathbf{y}^*$. Finding the value f^* attained by the optimal solution of (48) is complicated by the presence of the term $\|\boldsymbol{\mu}\|$. Therefore, we proceed as in the alternative approach of Section 4.2 and find the value \bar{f}^* (where $\bar{f}^* \geq f^*$ due to Theorem (4.5)) attained by the following problem

$$\begin{aligned} \max. \quad & \sum_{i=1}^m w_s \sigma(\tau_i) + w_q \mathbf{q}^T \mathbf{x}(\tau_i) + w_\mu \frac{\boldsymbol{\mu}^T \mathbf{x}(\tau_i)}{\|\bar{\boldsymbol{\mu}}\|} + \sum_{i=m+1}^n w_s \sigma(R[p_i]) + w_q \mathbf{q}^T \mathbf{y} + w_\mu \frac{\mathbf{y}^T \boldsymbol{\mu}}{\|\bar{\boldsymbol{\mu}}\|} \\ \text{s.t.} \quad & \mathbf{y}^T \mathbf{y} = 1 \end{aligned} \quad (49)$$

where $\bar{\boldsymbol{\mu}} = \frac{m}{n} \boldsymbol{\nu} + \frac{n-m}{n} \mathbf{q}$ is a constant obtained by considering a feasible solution $\bar{\mathbf{y}}_{m+1}, \dots, \bar{\mathbf{y}}_n$ that satisfies $\delta_i = 0$, i.e., $\bar{\mathbf{y}}_{m+1} = \dots = \bar{\mathbf{y}}_n = \mathbf{q}$. As shown in Appendix A.5, the optimal solution of (49) is given by

$$\bar{\mathbf{y}}_{m+1}^* = \dots = \bar{\mathbf{y}}_n^* = \frac{\mathbf{b}}{\sqrt{\mathbf{b}^T \mathbf{b}}} \quad (50)$$

where

$$\mathbf{b} = (n - m) \left[w_q \mathbf{q} + 2 \frac{w_\mu}{\|\bar{\boldsymbol{\mu}}\|} \frac{m}{n} \boldsymbol{\nu} \right] \quad (51)$$

6. PULLING STRATEGY

A pulling strategy identifies the relation R_i from which the next tuple is retrieved. We describe a *potential adaptive* (PA) strategy that is based on the current upper bounds t_M , defined for each set $M \subset \{1, \dots, n\}$. We generalize the approach introduced in [Finger and Polyzotis 2009] to the case of $n \geq 2$ relations. Let pot_i denote the *potential* of relation R_i , defined as $pot_i = \max\{t_M \mid M \subset \{1, \dots, n\} - \{i\}\}$, i.e., the upper bound on the aggregate score of combinations that can be formed with unseen tuples from R_i . The PA strategy is then defined as follows: access relation R_i such that pot_i is maximal (among pot_1, \dots, pot_n), breaking ties in favor of the relation with the least depth p_i , then the relation with the least index i .

Let *TBPA* and *TBRR* denote algorithms complying with the *ProxRJ* template using the tight bound (9) and, resp., the PA and round-robin strategy.

THEOREM 6.1. *Let I be a proximity rank join problem. Then $\text{depth}(TBPA, I, i) \leq \text{depth}(TBRR, I, i)$ for all i .*

PROOF. The proof of this theorem is very similar to the proof of Theorem 4.2 in [Schnaitter and Polyzotis 2008]. There, the authors show that their potential adaptive version of the *PBRJ* template with a corner bound ($PBRJ_c^*$) always terminates with a depth less than or equal to the depth of the round robin execution ($PBRJ_c^{RR}$), for every input relation. Their argument seamlessly adapts to our case by replacing *i*) $PBRJ_c^*$ with *TBPA*, *ii*) $PBRJ_c^{RR}$ with *TBRR*, and *iii*) their upper bound $\bar{S}(R_i(p_i))$ with the potential pot_i of relation R_i . The proof proceeds by contradiction, assuming an index k for which $\text{depth}(TBPA, I, k) > \text{depth}(TBRR, I, k)$. Let $p_i^{RR} = \text{depth}(TBRR, I, i)$ and let p_i be the depth of R_i when *TBPA* decides to pull $R_k[p_k^{RR} + 1]$. Then, it can be shown that, for each i , either *TBPA* has seen all tuples from R_i seen by *TBRR* or no tuple past $R_i[p_i]$ can participate in the solution. Thus, *TBPA* has already formed all combinations found by *TBRR*. Also, the score of the last combination found by *TBRR* is at least pot_i , for every i , and $\max\{pot_1, \dots, pot_n\}$ coincides with our tight bound. Therefore the termination condition of *TBPA* is met. Contradiction. \square

COROLLARY 6.2. *TBPA is instance optimal.*

PROOF. Trivial, by instance optimality of *TBRR* (Theorems 3.4 and 5.3) and Theorem 6.1. \square

7. EXPERIMENTAL STUDY

We investigate the following aspects: *i*) I/O cost reduction, in terms of the *sumDepths* metrics, that can be achieved using a tight bounding scheme and an adaptive pulling strategy, with respect to a simpler corner bound and a round-robin pulling strategy; *ii*) overhead, in terms of CPU time, due to the computation of a tight bound and potential savings that can be achieved when dominance is exploited; *iii*) impact, on the aforementioned metrics, of the problem parameters, i.e., number of results, dimensionality of the feature space, tuple density in the feature space, skewness of the tuple density, spatial skewness, number of relations, join condition and aggregation function weights.

7.1. Methodology

7.1.1. Data sets. First, we conduct our analysis on synthetic data whose relevant parameters are summarized in Table III. The data set is generated as follows.

For the experiments on Euclidean distance, for each of the relations R_i , $i = 1, \dots, n$, we generate a number of tuples. Each tuple τ_i is assigned a random score $\sigma(\tau_i)$, sampled from a uniform distribution, and a feature vector $\mathbf{x}(\tau_i)$. The feature vectors are obtained by sampling a d -dimensional uniform distribution centered in $\mathbf{0}$ a number of times so as to obtain the desired average density ρ expressed in terms of tuples per volume unit. When testing $n = 2$, we change the skewness parameter ρ_1/ρ_2 in such a way as to generate relations with different densities. Spatial skewness is achieved by sampling C vectors per volume unit, i.e., $\mathbf{c}_i \in \mathbb{R}^d$, $i = 1, \dots, C$. The feature vectors are sampled from a Gaussian Mixture Model (GMM) with C Gaussians with means \mathbf{c}_i and diagonal covariance matrix (variance $\sigma^2 = 0.0025$ along each dimension). When $C \rightarrow \rho$, the feature vectors are uniformly distributed. For low values of C , the feature vectors are distributed in C clusters, while preserving the overall density ρ .

Table III. Operating parameters (defaults in bold).

full name	parameter	tested values
Number of results	K	1, 10 , 50
Number of dimensions	d	1, 2 , 4, 8, 16
Density	ρ	20, 50, 100 , 200
Skewness	ρ_1/ρ_2	1, 2, 4, 8
Spatial skewness	C	100 , 8, 4, 2, 1
Number of relations	n	2 , 3, 4
Join condition	θ	∞ , 0.4, 0.2, 0.1, 0.05
Preference weight	w_q/w_s	10, 5, 1, 0.2, 0.1

For the experiments on cosine similarity, the feature vectors are obtained by sampling a uniform distribution on the unit d -dimensional hypersphere centered in 0 a number of times so as to obtain the desired average density ρ expressed in terms of tuples per surface unit.

For both Euclidean distance and cosine similarity, a join predicate might be added to select a subset of the combinations resulting from the cross-product. A range distance join predicated is used [Silva et al. 2010], whereby $\tau = \tau_1 \times \dots \times \tau_n$ is selected if $\delta(\mathbf{x}(\tau_i), \mathbf{x}(\tau_j)) \leq \theta, \forall i, j$.

We emphasize that the size of the data sets is not a relevant parameter in our study. Indeed, solving the proximity rank join problem for a target number of results K calls for retrieving only a prefix of the relations.

Then, we test all methods on two real data sets. For the case of Euclidean distance, we consider relations containing hotels, theaters, and restaurants in five different American cities (San Francisco, New York, Boston, Dallas and Honolulu). For each such city, we test the case described in Example 2.1 in Section 1. Each data set is obtained by retrieving customer ratings, latitude and longitude (thus $d = 2$) of entertainment places in all five cities by means of the YQL console available at [YQL]. These data sets are used to feed $n = 3$ Web services endowed with distance-based access returning, respectively, hotels, restaurants and cinemas. The query vector \mathbf{q} is represented by a specific location within these cities (e.g., Fishermans Wharf in San Francisco, Battery Park in New York, etc.). For each query, we retrieve the top-10 combinations.

For the case of cosine similarity, we consider relations containing annotation profiles of genes of two different organisms largely studied in the field of bioinformatics: *Saccharomyces cerevisiae* (SGD) and *Drosophila melanogaster* (FlyBase). Each gene is annotated with terms defining its biological process attributes by means of the Gene Ontology [Consortium 2001]. Annotation profiles are stored as a bit vectors whose size equals the number of terms in the vocabulary (the bit is set to 1 if the term is used to annotate the gene, 0 otherwise). We restricted the vocabulary to terms used to annotate (after unfolding) at least three genes in each organism, resulting in 346 terms. The annotation profile of each gene is then projected to a 8-dimensional subspace by means of latent semantic analysis [Khatri et al. 2005], which handles correlation between terms, as learned by analyzing co-occurrence of terms in the dataset.

Our test queries aim to discover orthologous genes from the two organisms given a target annotation profile, i.e., to extract the genes that are similar to a given profile as well as to each other. For that, we sampled the annotation profiles of five genes at random. The aggregation function in use is as in (4), with $w_q = 1$, $w_\mu = 0.1$, $w_s = 1$ (however, w_s is immaterial to ranking, as genes do not have an associated score).

7.1.2. Methods. We test different instantiations of the *ProxRJ* template described in Algorithm 1. These are the result of combining the different bounding schemes with the two pulling strategies. We denote the tested algorithms as *CBRR* (Corner Bound, Round Robin), *CBPA* (Corner Bound, Potential Adaptive), *TBRR* (Tight Bound, Round

Robin), and *TBPA* (Tight Bound, Potential Adaptive). For the case of cosine similarity, our proposed approach uses a bound obtained by solving (35) exactly, which provides a correct upper bound that is not always the tight bound, although very close to it. For this reason, we refer to it as *good bound* and test the two corresponding algorithms *GBRR* (Good Bound, Round Robin) and *GBPA* (Good Bound, Potential Adaptive) in our experiments. Also, we observe that *CBRR* and *CBPA* correspond to, resp., *HRJN* and *HRJN** [Ilyas et al. 2004]. The dominance test is always performed for both distance-based and score-based access.

7.1.3. Evaluation metrics. We adopt *sumDepths* as the primary metrics for comparing the different algorithms. This is especially relevant in application scenarios where the cost of fetching tuples from the relations largely dominates over computing the combinations and their respective bounds. This is the case, e.g., of search services invoked over the Web. We also report the *total CPU time*, in seconds, for the various experiments. We do not measure the time needed for fetching the tuples, as this is implicitly captured by the *sumDepths* metrics. Moreover, we indicate the fraction of time consumed by the calls to the function *updateBound*. For fairness, we compute both metrics over ten different data sets and report the average.

7.2. Testing environment.

All the algorithms have been executed on a PC with the Windows 7 operating system, an Intel[®] Core2-Duo processor at 2.4GHz and 4Gb RAM. We remark that the *sumDepth* metrics is completely oblivious of the testing environment. As for the CPU time, we measured the wall clock execution time needed to return the top-*K* combinations, assuming that tuples of the joined relations are available locally in main memory. Thus, we did not consider the time needed for fetching the tuples when data is available from remote sources.

7.3. Results

The results obtained are summarized in Figures 8-24. Each figure refers to either distance-based or score-based access, and reports the results for both Euclidean distance and cosine similarity. In the stacked bar charts reporting the total CPU time, *i*) the darker bars at the bottom represent the cost of forming the combinations and computing their aggregate score, and *ii*) the lighter bars on top represent the cost of executing *updateBound*. As a general trend, we observe that the use of a tight (or good) bounding always outperforms the simpler corner bound in terms of the *sumDepths* metrics by a noticeable margin, i.e., at least 15% in the worst case. This comes at the cost of an increased complexity when computing the bound. However, the average total CPU time in our experiments, for $n = 2$, is in most cases less than 0.5 seconds. Thus, in a scenario where data is accessed by means of remote service invocations, the *total* CPU time is of the same order of magnitude as the time needed for a *single* access. Moreover, we note that in our experiments the tight (or good) bound is recomputed after every accessed tuple. While this guarantees to access the minimum number of tuples, in practical systems a good trade-off can be achieved by recomputing the tight bound only after retrieving blocks of tuples.

For the case of cosine similarity, we do not show the results for *TBRR* and *TBPA*, as the improvement in terms of *sumDepths* over, resp., *GBRR* and *GBPA* is negligible (at most 2 units in all our experiments). In our tests, we computed the tight bound by extensive grid search with very high accuracy, which required an effort in terms of CPU time several orders of magnitude higher than with the other algorithms. Note also that the good bound used by *GBRR* and *GBPA* is constructed in such a way that it coincides with the actual tight bound in case the solution of (34) satisfies the constraints with

the equality, which is likely to be the case in practice whenever the weight w_q is heavier than w_μ .

We now comment on the effect of all the individual parameters.

7.3.1. Number of results - K . As in conventional rank join problems, the number of results grows sublinearly with respect to the number of accessed tuples (Figures 8(a), 8(b), 9(a), and 9(b)). The gain of *TBPA* over *CBPA* is between 25% and 45% and is larger for smaller values of K . The adaptive pulling strategy reduces the number of accessed tuples by 5-10%, although the two relations contain data sampled from identical distributions. *TBPA* requires approximately 4 times more CPU time than *CBPA* with distance-based access for Euclidean distance (Figure 8(c)), while only twice as much for cosine similarity (Figure 8(d)); no extra burden is measured in the case of score-based access (Figure 9(c) and 9(b)), as computing the bound is much more efficient due to the aggressive pruning obtained by using dominance in this case.

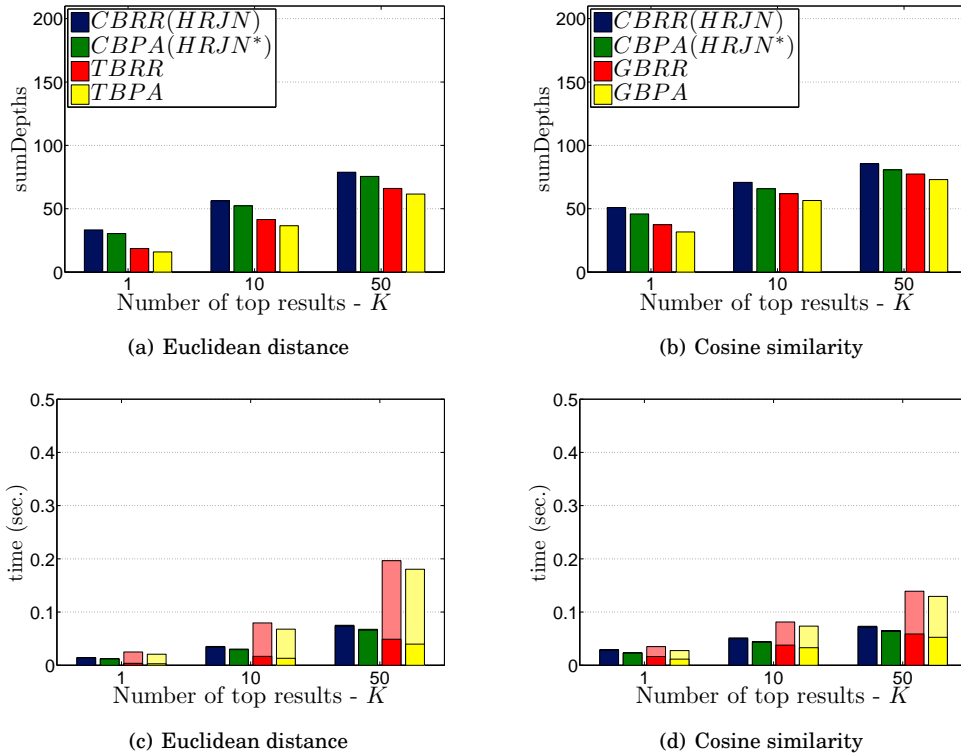


Fig. 8. Performance of the tested algorithms as a function of the number of results K for *distance-based access*.

7.3.2. Number of dimensions - d . This is a characteristic parameter of the proximity rank join problem. For Euclidean distance with distance-based access (Figure 10(a)), the gain of *TBPA* over *CBPA* is between 15% and 45% and it is larger in higher dimensional spaces. Indeed, for the same density per volume unit, higher dimensional spaces are emptier, in the sense that the average inter-tuple distance is larger. This fact is neglected by the simpler corner bounding scheme adopted by *CBRR* and *CBPA*, which always assume a zero distance from the centroid for unseen combinations. Note that

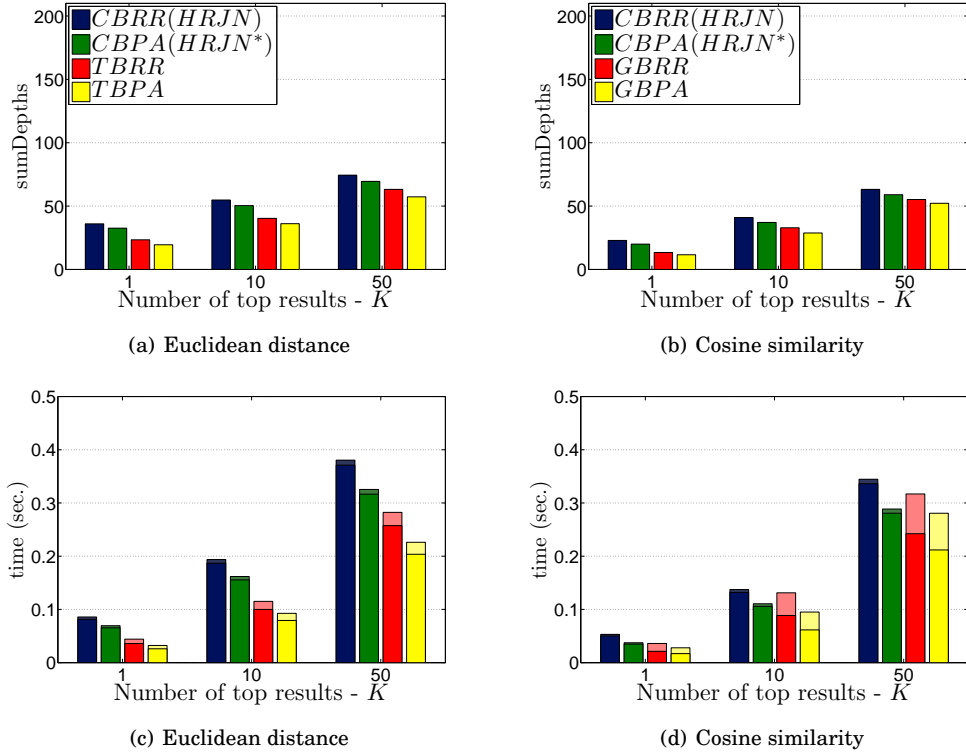


Fig. 9. Performance of the tested algorithms as a function of the number of results K for *score-based access*.

the total CPU time (Figure 10(c)) scales favorably when d increases. This is due to two facts: first, the number of accessed tuples decreases as d increases, thus computing the tight bound needs to consider fewer partial combinations; second, for the same number of accessed tuples, the total CPU time does not depend on d , since computing the upper bound on the score of a partial combination requires solving a 1-dimensional problem, regardless of d . Similar considerations and performance gains hold for the case of cosine similarity (Figures 10(b) and 10(d)). A different trend is observed in Figures 11(a) and 11(c) for the case of score-based access, as the *sumDepths* metrics increases when d increases for all tested algorithm, although relative gains due to the tight bounding scheme remain the same. Here, the upper bound used to terminate the algorithm is independent of d , as it depends mostly on the scores. At the same time, a larger average inter-tuple distance induces lower average values of the aggregate scores, according to (3). Thus, for the same depths, fewer combinations have an aggregate score exceeding the upper bound. This is not the case for distance-based access in Figure 10(a), since the bounding scheme inherently takes into account the geometry of the space and the corresponding inter-tuple distances when computing the upper bound.

7.3.3. Density - ρ (number of tuples per volume unit). *TBPA* and *CBPA* are similarly affected by the tuple density, and the gain of the former is always in the 20-30% range. The *sumDepths* metrics increases with the density (Figure 12(a), 12(b), 13(a) and 13(b)). This is due to the higher probability of generating combinations with a large aggregate score by means of random sampling.

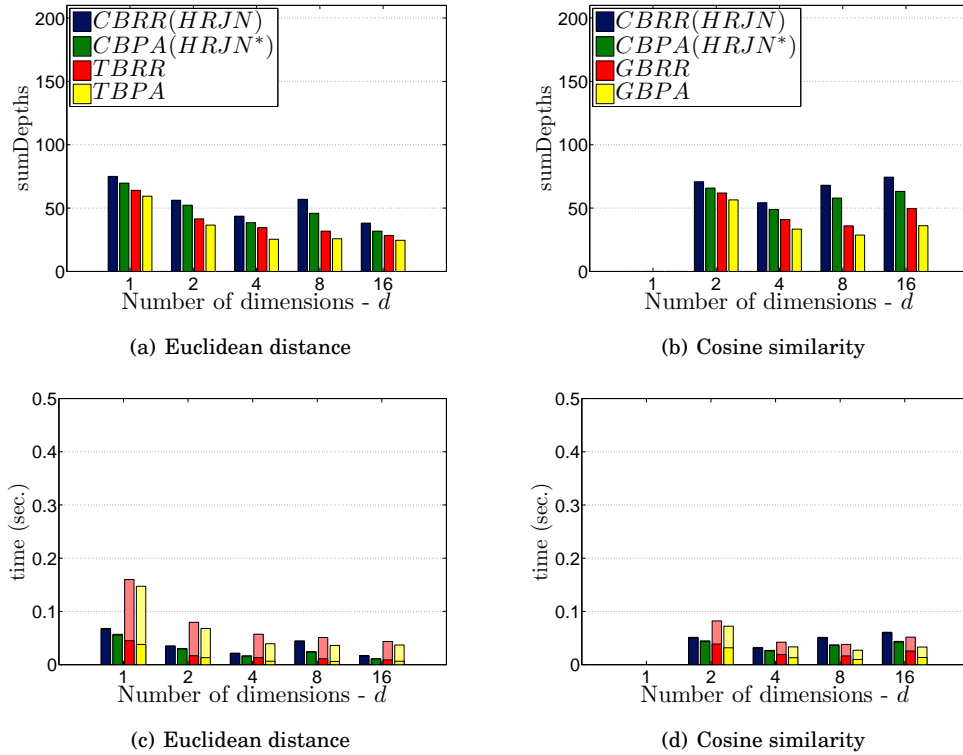


Fig. 10. Performance of the tested algorithms as a function of the number of dimensions d for *distance-based access*.

7.3.4. Skewness - ρ_1/ρ_2 (ratio of densities for two relations). Generating skewed data sets highlights the benefits of an adaptive pulling strategy. The gain over round-robin can be as large as 25-30% when $\rho_1/\rho_2 \geq 4$ (Figure 14(a), 14(b), 15(a) and 15(b)).

7.3.5. Spatial skewness - C . With spatially skewed data sets, the query vector might fall in a region which is not densely populated. Therefore, the proximity weighted scores of top- K combinations decrease with C , thus *sumDepths* increases (Figures 16(a), 16(b), 17(a) and 17(b)). This trend is more pronounced for distance-based access. *TBPA* and *CBPA* are similarly affected by the tuple density, and the gain of the former is always in the 20-30% range.

7.3.6. Number of relations - n . The gain of *TBPA* over *CBPA* is significantly larger when joining more than two relations, attaining more than 50% for $n = 3$. We observe that, in this case, *TBPA* outperforms *CBPA* in terms of both *sumDepths* (Figure 18(a), 18(b), 19(a) and 19(b)) and total CPU time (Figure 18(c), 18(d), 19(c) and 19(d)). Indeed, the additional computational burden related to the tight bounding scheme is more than compensated by the significantly smaller number of combinations (approximately 4000 as opposed to 32000 for the case of distance-based access with Euclidean distance) that need to be formed. This observation is further confirmed for $n = 4$. In this case, *CBPA* was unable to report the top-10 results (within five minutes) when adopting distance-based access, due to the extremely large number of combinations that need to be formed.

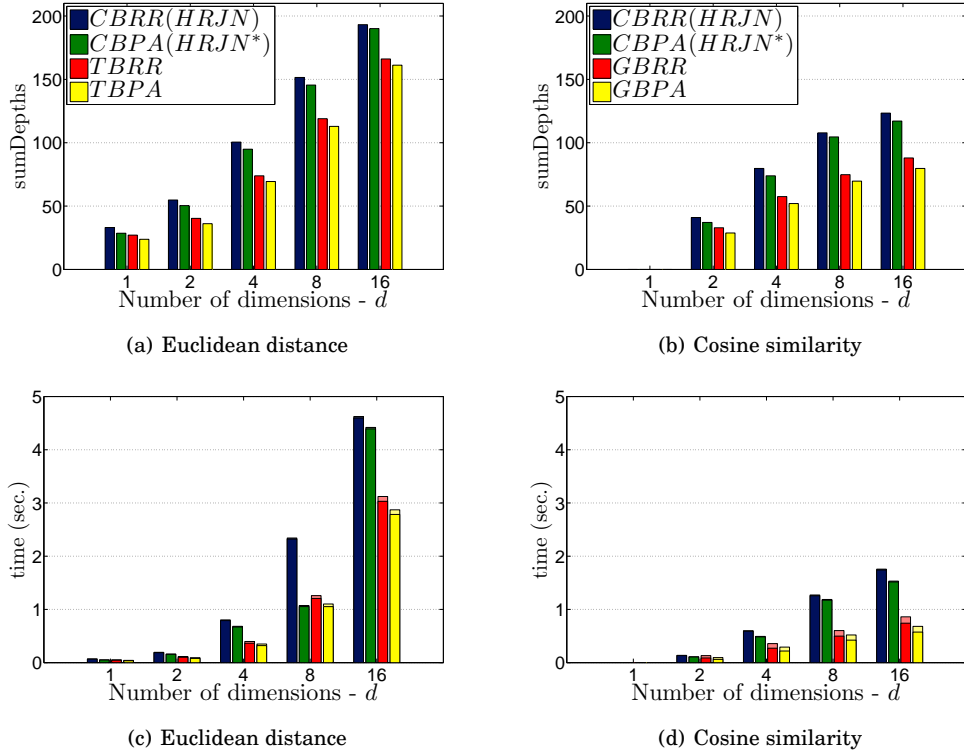


Fig. 11. Performance of the tested algorithms as a function of the number of dimensions d for *score-based access*.

7.3.7. Join condition - θ . Although the *ProxRJ* template focuses on cross-product, it can be straightforwardly adapted to the case in which a join predicate is added. By varying the value of parameter θ , it is possible to set join conditions targeting different join selectivity levels. Specifically, when $\theta \rightarrow \infty$, all join results are selected, while fewer join results are selected by decreasing θ . For fairness, we evaluate the impact of θ when $n = 3$ relations are joined, since when $n = 2$ the total CPU time is not affected: all combinations in the cartesian product are visited anyway, and those that do not satisfy the join predicate are then discarded, while for $n > 2$ some combinations may not even be visited. For large values of θ , the join results that are discarded are those combinations whose tuples are far apart from each other, which are unlikely to be reported among the top- K results. Hence, in Figures 20(a) and 21(a), we observe that the *sumDepths* metrics is not affected. On the other hand, the total CPU time (Figures 20(c) and 21(c)) decreases due to the fact that: *i*) fewer full combinations are formed; *ii*) fewer partial combinations are formed, for which the upper bound needs to be computed. For small values of θ , combinations with a high proximity-weighted score might be discarded, thus a larger number of tuples needs to be retrieved to find the top- K join results, and the *sumDepths* metrics increases. Similar results are obtained for the case of cosine similarity (Figures 20(b) and 21(b)), where the join condition is expressed in terms of the angle between two feature vectors for the sake of clarity.

7.3.8. Aggregation function weights - w_q/w_s . In some circumstances, both distance-based and score-based access might be available. Which access kind should be preferred

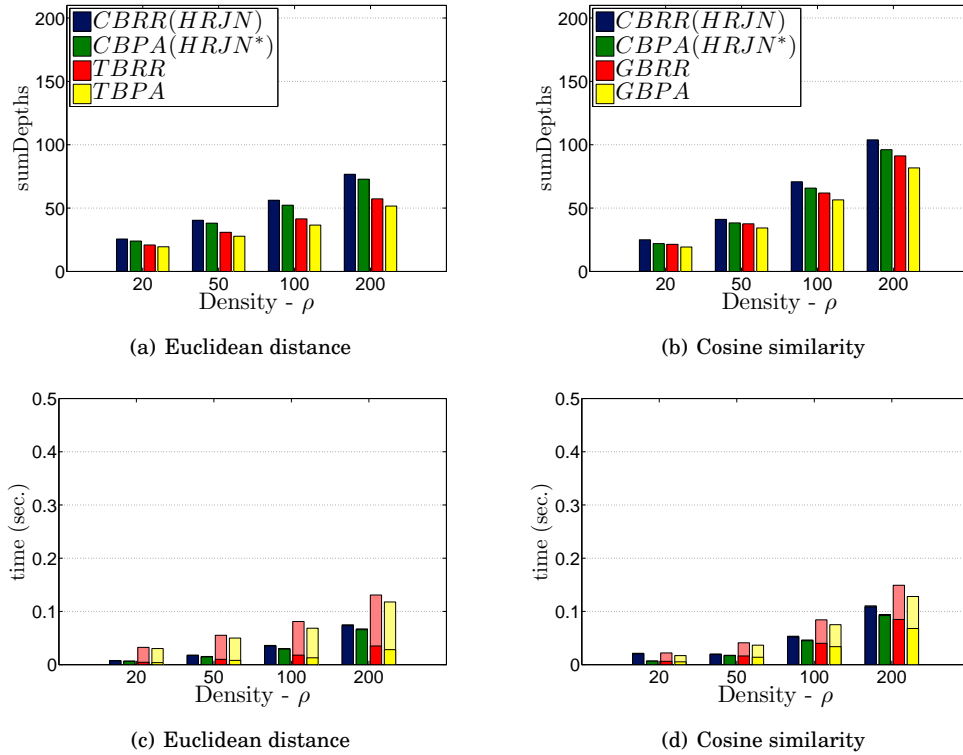


Fig. 12. Performance of the tested algorithms as a function of tuple density ρ for *distance-based access*.

largely depends on the preference weights w_q and w_s . Intuitively, when the ratio w_q/w_s increases, distance-based access leads to fewer retrieved tuples, while the opposite holds for score-based access. This is confirmed by Figures 22(a) and 23(a), where the ratio w_q/w_s is varied, while keeping $w_\mu = w_s$. In all cases, *TBPA* (*GBPA*) always outperforms *CBPA* (*CBPA*).

7.3.9. Dominance. The dominance test described in Section 3.2.1 identifies those partial combinations that cannot contribute to the top- K join results, so that their upper bound need not be updated. As such, dominance does not affect the *sumDepths* metrics, but it impacts the total CPU time, reducing the time needed to update the upper bound. Figures 24(a) and 24(b) show that the total CPU time is halved when the dominance test is enabled regardless of the number of top results, for both Euclidean distance and cosine similarity. Figures 24(c) and 24(d) analyze the impact of dominance when joining more than two relations. We observe an n -fold decrease in the fraction of time needed to update the upper bound.

7.3.10. Real data sets. Finally, Figures 25 and 26 report the results obtained on the real data sets. For the case of Euclidean distance and distance-based access, Figure 25(a) shows that *TBPA* outperforms *CBPA* by a large margin (on average, 50%). The adaptive pulling strategy is always beneficial, regardless of the bounding scheme, reducing by 10-20% the number of accessed tuples. For the case of score-based access, Figure 25(b) shows that a larger number of tuples needs to be retrieved to obtain the top-10 combinations than with distance-based access. This is due to the fact that many

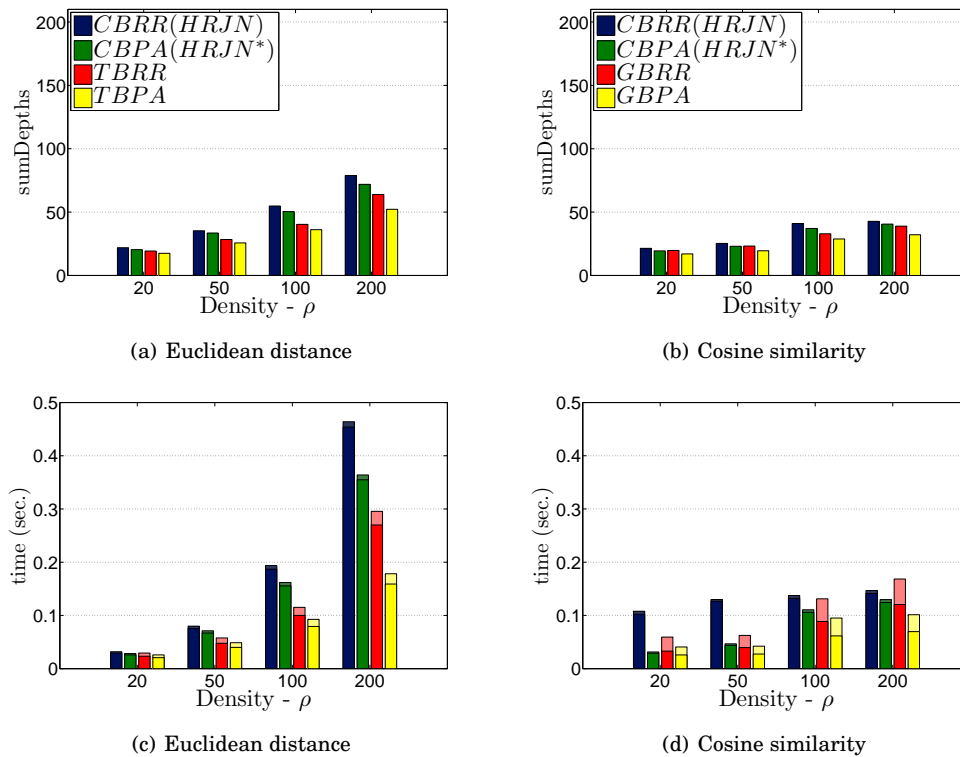


Fig. 13. Performance of the tested algorithms as a function of tuple density ρ for *score-based access*

tuples in each relation are assigned the largest possible score value. Thus, the upper bound does not decrease significantly until all such tuples are retrieved. However, *TBPA* always outperforms *CBPA* (on average 15%).

Similar results are obtained in the case of cosine similarity, as illustrated in Figure 26(a). There, *GBPA* outperforms *CBPA* by 50-70%, depending on the query.

8. RELATED WORK

Proximity rank join is a significant extension of rank join, a class of problems recently discussed in [Ilyas et al. 2004; Schnaitter et al. 2007; Schnaitter and Polyzotis 2008; Finger and Polyzotis 2009; Ilyas et al. 2008].

State-of-the-art rank join algorithms [Ilyas et al. 2004; Schnaitter and Polyzotis 2008; Finger and Polyzotis 2009] inherit the idea of using an upper bound from the threshold-based stopping condition of the well-known *threshold algorithm* [Fagin et al. 2003] (TA). TA addresses rank aggregation, which is the problem of combining several ranked lists into a single consensus ranking.

Our main starting point is [Schnaitter and Polyzotis 2008], where the rank join *PBRJ* template is introduced, which we have specialized and extended into *ProxRJ* for proximity rank join. This template encompasses the well-known *HRJN* and *HRJN** operators [Ilyas et al. 2004], corresponding to our *CBRR* and *CBPA* algorithms. In [Schnaitter and Polyzotis 2008], rank join is studied in the case where each relation may be equipped with even more than one score attribute, while we have focused on relations with a single score. As was discussed in Section 3, the results in [Schnaitter

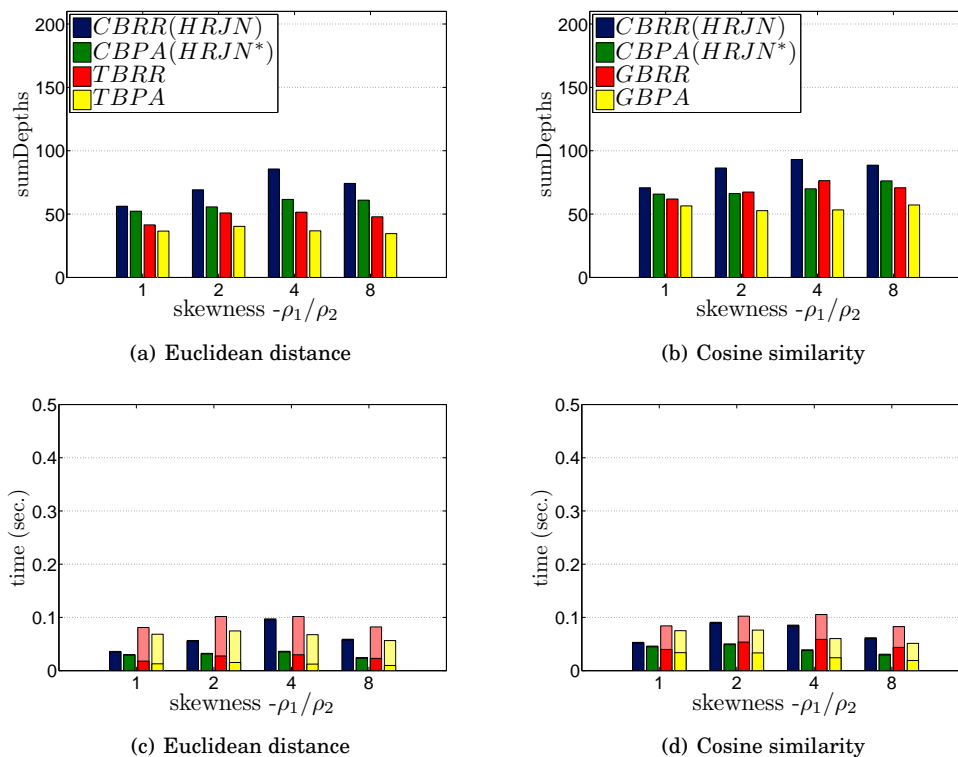


Fig. 14. Performance of the tested algorithms as a function of skewness ρ_1/ρ_2 for *distance-based access*.

and Polyzotis 2008] do not always carry over to proximity rank join, due to the geometry of the problem (absent in rank join). These include, e.g., the fact that *HRJN* is instance optimal with two relations (with a single score) for rank join but not for proximity rank join.

In [Finger and Polyzotis 2009], the authors propose an efficient way for computing an approximation of the tight bound, in order to find a good trade off between I/O cost and CPU cost.

In [Agrawal and Widom 2009], the authors study (two-way) rank join problems, in which memory availability is limited, and propose algorithms that are instance-optimal also with this restriction.

We have considered a scenario in which the objects' feature vectors are deterministic. Others have addressed related problems (e.g., nearest neighbors) when the objects have uncertain features, such as approximate positions [Beskales et al. 2008].

Weighted proximity join is introduced in [Thonangi et al. 2009], where algorithms are proposed to find combinations of words in documents that appear close to each other and match different query terms. Their approach differs from ours in the following aspects: *i)* the aggregation function does not depend on the distance from the query, and all the input needs to be read to find the best combinations; *ii)* the proposed algorithms are specifically tailored to 1-dimensional spaces. Similarity join is promoted to a first-class operator in [Silva et al. 2010].

This work builds upon [Martinenghi and Tagliasacchi 2010], where the authors introduce proximity rank join focusing on a specific case of Euclidean distance. Here, we

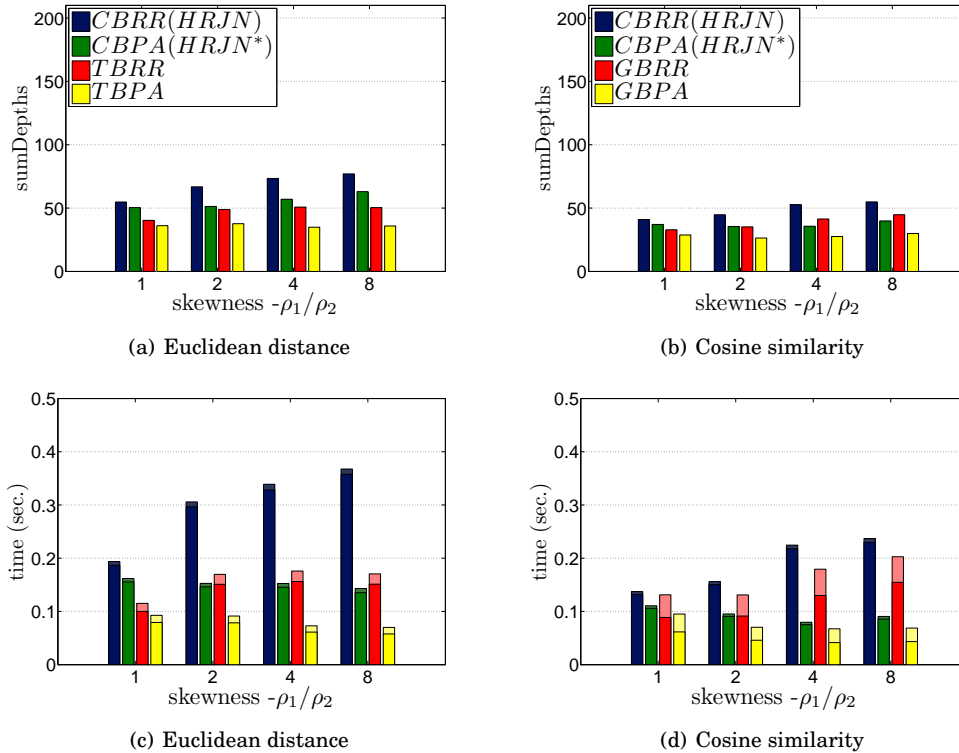


Fig. 15. Performance of the tested algorithms as a function of skewness ρ_1/ρ_2 for *score-based access*.

also study the problem for a proximity measure that is not a metric, as is the case with cosine similarity. While such a measure fits the general framework described in Section 3, this required to devise entirely new techniques for efficiently computing the tight bounds (Sections 4.2 and 5.2). In addition, we have significantly extended our experimental evaluation to also cover the cases with cosine similarity and with score-based access.

The study of join predicates that depend on the spatial proximity among the objects has been thoroughly investigated in the past literature. The work in [Hjaltason and Samet 1998] presents incremental distance joins, e.g., how to compute the spatial join between two lists of objects and reporting the closest pairs early. A non-incremental algorithm for the K-closest-pair problem is investigated in [Corral et al. 2004]. This is extended in [Papadopoulos et al. 2006], where additional spatial constraints are imposed on the domain of the objects belonging to the two lists (e.g., all points of the first list that are part of the answer are enclosed in a given region). The top-k spatial join described in [Mamoulis et al. 2005] is a binary join operator that returns, for each list, the object with the largest number of overlapping objects in the other list, thus extending the previous approaches to objects covering finite regions. Nevertheless, in all of the above-mentioned works, the authors assume that input relations are indexed by structures of the R-tree family. This contrasts our setting, where indexes are unavailable and relations are accessed either by distance or by score.

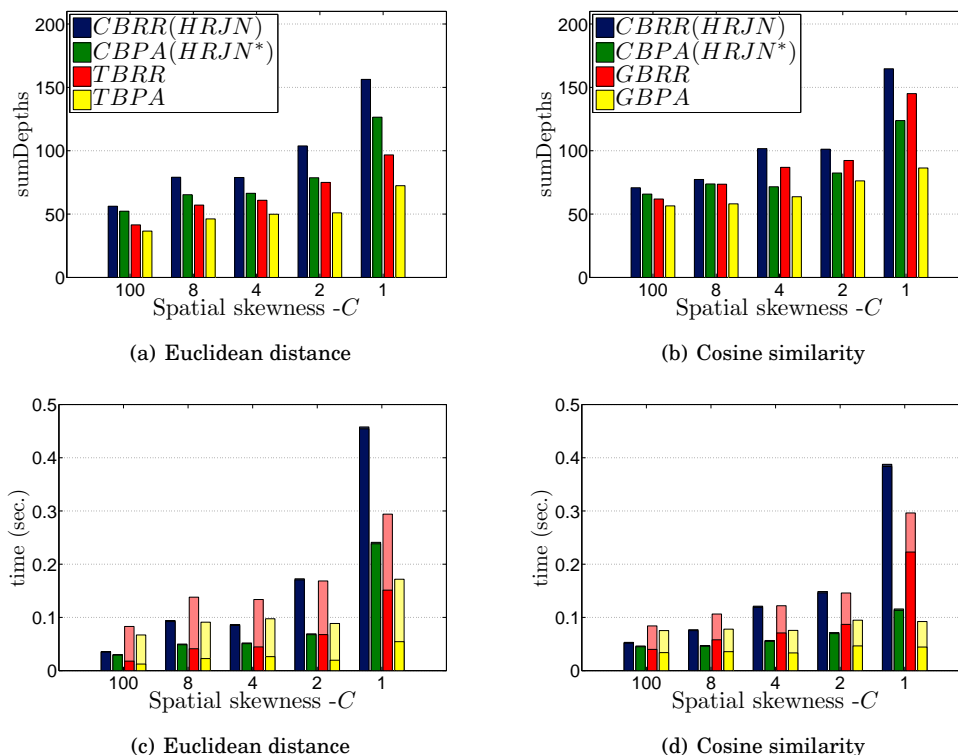


Fig. 16. Performance of the tested algorithms as a function of spatial skewness C for *distance-based access*.

9. CONCLUSION

We introduced proximity rank-join as the problem of finding the best combinations of heterogeneous objects that are close to a given target object (the query) and to each other. We gave a general formulation of the problem and proposed an instance optimal algorithm that solves it based on the computation of a tight upper bound on the aggregate score of the combinations to be formed. Our solution proves superior to existing rank join approaches, also experimentally.

We plan to extend proximity rank join to the case of relations that can be accessed not only by sorted access but also by random access.

REFERENCES

- YQL console. <http://developer.yahoo.com/yql/console/>.
- AGRAWAL, P. AND WIDOM, J. 2009. Confidence-aware join algorithms. In *ICDE*. 628–639.
- BESKALES, G., SOLIMAN, M. A., AND ILYAS, I. F. 2008. Efficient search for the top-k probable nearest neighbors in uncertain databases. *PVLDB I*, 1, 326–339.
- CONSORTIUM, T. G. O. 2001. Creating the gene ontology resource: Design and implementation. *Genome Research* 11, 8, 1425–1433.
- CORRAL, A., MANOLOPOULOS, Y., THEODORIDIS, Y., AND VASSILAKOPOULOS, M. 2004. Algorithms for processing k-closest-pair queries in spatial databases. *Data Knowl. Eng.* 49, 1, 67–104.
- FAGIN, R., LOTEM, A., AND NAOR, M. 2003. Optimal aggregation algorithms for middleware. *J. Comput. Syst. Sci.* 66, 4, 614–656.
- FINGER, J. AND POLYZOTIS, N. 2009. Robust and efficient algorithms for rank join evaluation. In *SIGMOD Conference*. 415–428.

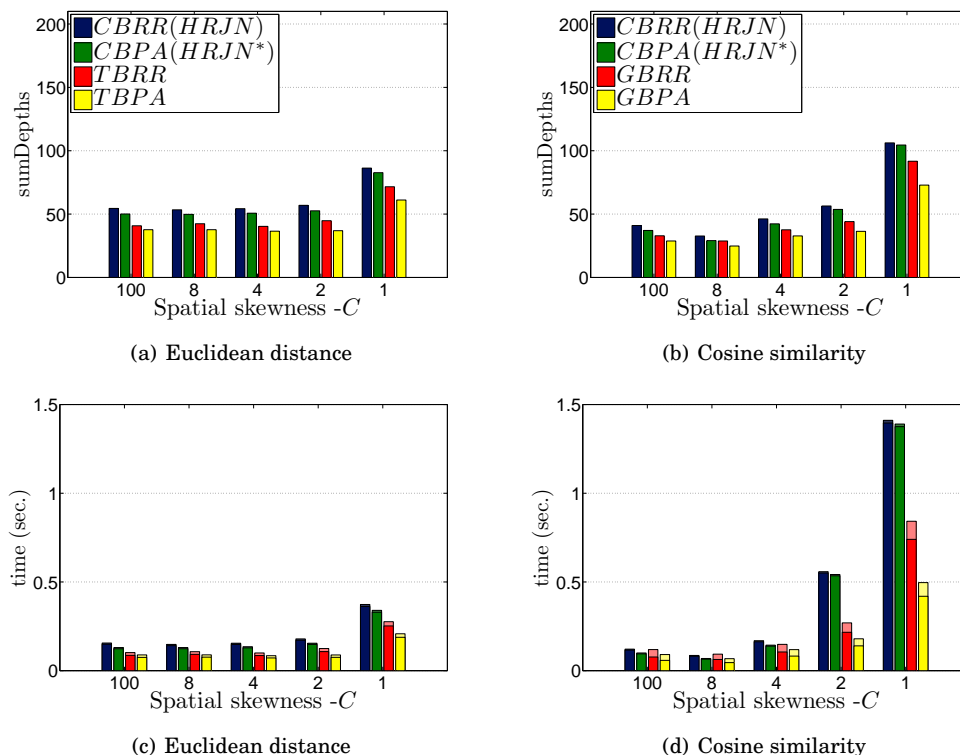


Fig. 17. Performance of the tested algorithms as a function of spatial skewness C for *score-based access*.

- HJALTASON, G. R. AND SAMET, H. 1998. Incremental distance join algorithms for spatial databases. In *SIGMOD Conference*. 237–248.
- ILYAS, I. F., AREF, W. G., AND ELMAGARMID, A. K. 2004. Supporting top-k join queries in relational databases. *VLDB J.* 13, 3, 207–221.
- ILYAS, I. F., BESKALES, G., AND SOLIMAN, M. A. 2008. A survey of top- query processing techniques in relational database systems. *ACM Comput. Surv.* 40, 4.
- KHATRI, P., DONE, B., RAO, A., DONE, A., AND DRAGHICI, S. 2005. A semantic analysis of the annotations of the human genome. *Bioinformatics* 21, 16, 3416–3421.
- MAMOULIS, N., THEODORIDIS, Y., AND PAPADIAS, D. 2005. Spatial joins: Algorithms, cost models and optimization techniques. In *Spatial Databases*. 155–184.
- MARTINENGI, D. AND TAGLIASACCHI, M. 2010. Proximity rank join. *PVLDB* 3, 1, 352–363.
- PAPADOPOULOS, A. N., NANOPOULOS, A., AND MANOLOPOULOS, Y. 2006. Processing distance join queries with constraints. *Comput. J.* 49, 3, 281–296.
- SCHNAITTER, K. AND POLYZOTIS, N. 2008. Evaluating rank joins with optimal cost. In *PODS*. 43–52.
- SCHNAITTER, K., SPIEGEL, J., AND POLYZOTIS, N. 2007. Depth estimation for ranking query optimization. In *VLDB*. 902–913.
- SILVA, Y. N., AREF, W. G., AND ALI, M. H. 2010. The similarity join database operator. In *ICDE*.
- THONANGI, R., HE, H., DOAN, A., WANG, H., AND YANG, J. 2009. Weighted proximity best-joins for information retrieval. In *ICDE*. 234–245.

A. DERIVATIONS

A.1. Solution of (15)

We assume w.l.o.g. that $\mathbf{q} = \mathbf{0}$ or, equivalently, that all the feature vectors are referred to a coordinate system that has the origin in \mathbf{q} , i.e., $\mathbf{x}_i \leftarrow \mathbf{x}_i - \mathbf{q}$, for $i = 1, \dots, m$, and we

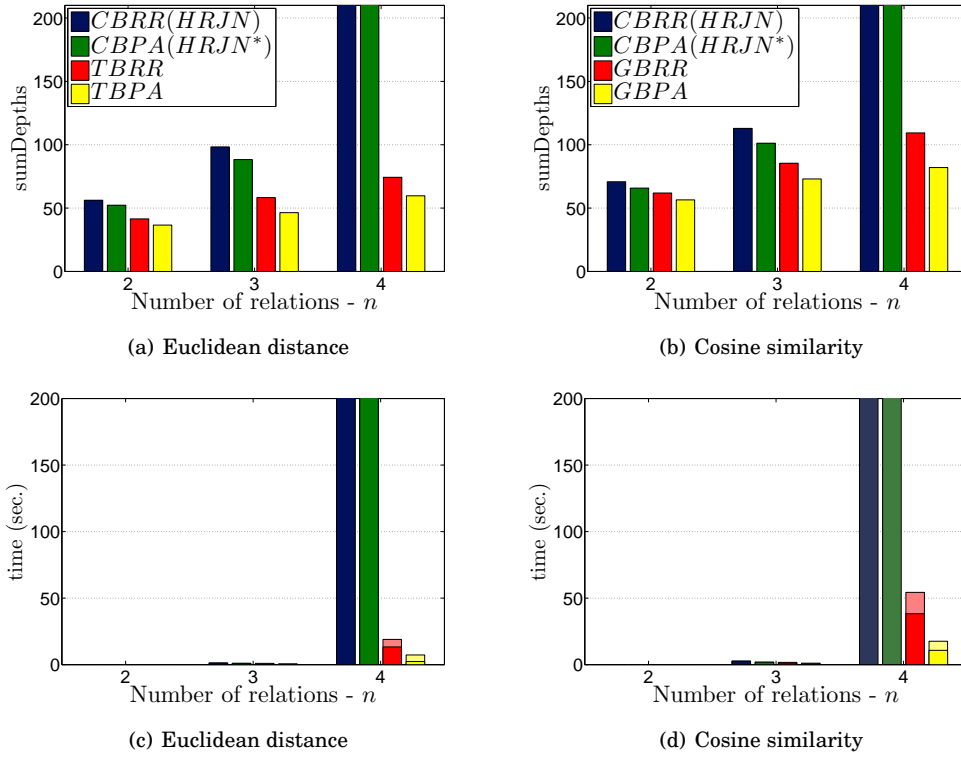


Fig. 18. Performance of the tested algorithms as a function of the number of relations n for distance-based access.

adopt the shorthand notation $\mathbf{x}_i = \mathbf{x}(\tau_i)$. The centroid $\boldsymbol{\mu}$ of the full combination can be written as

$$\boldsymbol{\mu} = \frac{1}{n} \left(\sum_{i=1}^m \mathbf{x}_i + (n-m)\mathbf{y} \right) = \frac{1}{n} \sum_{i=1}^m \mathbf{x}_i + \frac{n-m}{n} \mathbf{y} = \frac{m}{n} \boldsymbol{\nu} + \frac{n-m}{n} \mathbf{y} \quad (52)$$

where $\mathbf{y} = \mathbf{y}_{m+1} = \dots = \mathbf{y}_n$ due to the symmetry of the problem, and $\boldsymbol{\nu} = \frac{1}{m} \sum_{i=1}^m \mathbf{x}_i$ is the centroid of the partial combination.

The objective function of (15) is

$$\max. \sum_{i=1}^m [w_s \ln(\sigma(\tau_i)) - w_q \|\mathbf{x}_i\|^2 - w_\mu \|\mathbf{x}_i - \boldsymbol{\mu}\|^2] + \sum_{i=m+1}^n [w_s \ln(\sigma_i^{\max}) - w_q \|\mathbf{y}\|^2 - w_\mu \|\mathbf{y} - \boldsymbol{\mu}\|^2],$$

which can be rewritten as follows:

$$\min. \quad w_q(n-m)\|\mathbf{y}\|^2 + w_\mu(n-m)\|\mathbf{y} - \boldsymbol{\mu}\|^2 + w_\mu \sum_{i=1}^m \|\mathbf{x}_i - \boldsymbol{\mu}\|^2 + w_q \sum_{i=1}^m \|\mathbf{x}_i\|^2 - w_s \sum_{i=1}^m \ln(\sigma(\tau_i)) - w_s \sum_{i=m+1}^n \ln(\sigma_i^{\max}), \quad \text{i.e.,}$$

$$\min. \quad w_q(n-m)\mathbf{y}^T \mathbf{y} + w_\mu(n-m)(\mathbf{y} - \boldsymbol{\mu})^T (\mathbf{y} - \boldsymbol{\mu}) + w_\mu \sum_{i=1}^m (\mathbf{x}_i - \boldsymbol{\mu})^T (\mathbf{x}_i - \boldsymbol{\mu}) + s, \quad (53)$$

where s is a scalar defined as

$$s = w_q \sum_{i=1}^m \|\mathbf{x}_i\|^2 - w_s \sum_{i=1}^m \ln(\sigma(\tau_i)) - w_s \sum_{i=m+1}^n \ln(\sigma_i^{\max}), \quad (54)$$

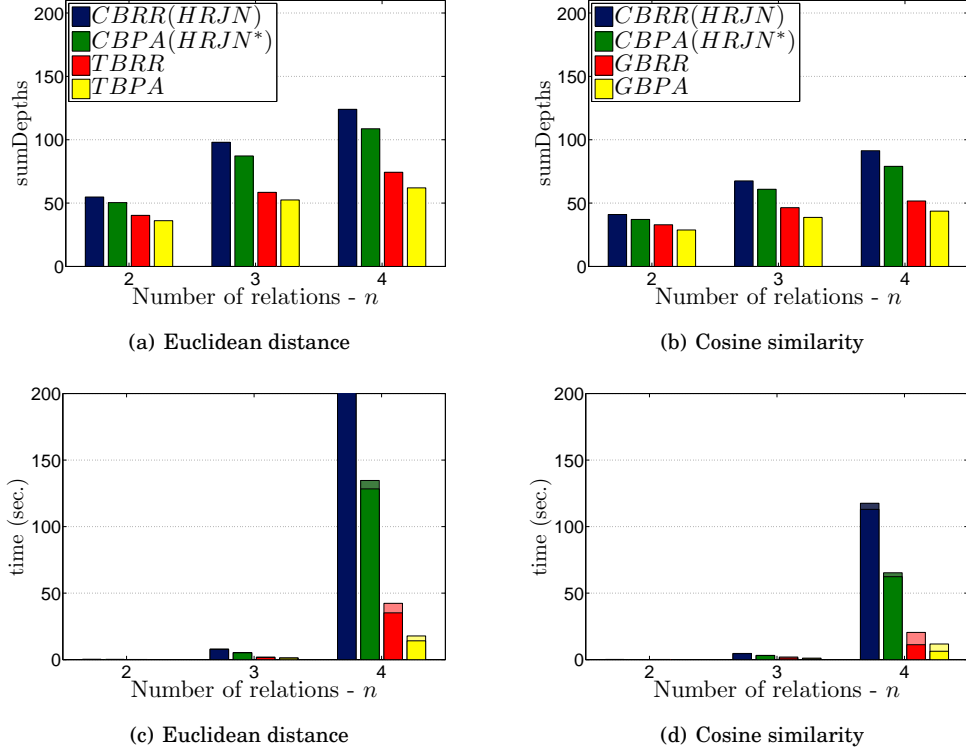


Fig. 19. Performance of the tested algorithms as a function of the number of relations n for *score-based access*.

which does not depend on \mathbf{y} . In turn, (53) can be rewritten as follows.

$$\min. \quad w_q(n-m)\mathbf{y}^T\mathbf{y} + w_\mu(n-m)\left(\mathbf{y} - \frac{m}{n}\boldsymbol{\nu} - \frac{n-m}{n}\mathbf{y}\right)^T\left(\mathbf{y} - \frac{m}{n}\boldsymbol{\nu} - \frac{n-m}{n}\mathbf{y}\right) + w_\mu \sum_{i=1}^m \left(\mathbf{x}_i - \frac{m}{n}\boldsymbol{\nu} - \frac{n-m}{n}\mathbf{y}\right)^T\left(\mathbf{x}_i - \frac{m}{n}\boldsymbol{\nu} - \frac{n-m}{n}\mathbf{y}\right) + s, \quad \text{i.e.,}$$

$$\min. \quad w_q(n-m)\mathbf{y}^T\mathbf{y} + w_\mu(n-m)\left(\frac{m}{n}\mathbf{y} - \frac{m}{n}\boldsymbol{\nu}\right)^T\left(\frac{m}{n}\mathbf{y} - \frac{m}{n}\boldsymbol{\nu}\right) + w_\mu \sum_{i=1}^m \left[\left(\mathbf{x}_i - \frac{m}{n}\boldsymbol{\nu}\right) - \frac{n-m}{n}\mathbf{y}\right]^T\left[\left(\mathbf{x}_i - \frac{m}{n}\boldsymbol{\nu}\right) - \frac{n-m}{n}\mathbf{y}\right] + s, \quad \text{i.e.,}$$

$$\min. \quad w_q(n-m)\mathbf{y}^T\mathbf{y} + w_\mu(n-m)\frac{m^2}{n^2}\mathbf{y}^T\mathbf{y} + w_\mu(n-m)\frac{m^2}{n^2}\boldsymbol{\nu}^T\boldsymbol{\nu} - 2w_\mu(n-m)\frac{m^2}{n^2}\boldsymbol{\nu}^T\mathbf{y} + w_\mu \sum_{i=1}^m \left[\left(\mathbf{x}_i - \frac{m}{n}\boldsymbol{\nu}\right)^T\left(\mathbf{x}_i - \frac{m}{n}\boldsymbol{\nu}\right) + \left(\frac{m-n}{n}\right)^2\mathbf{y}^T\mathbf{y} - 2\frac{n-m}{n}\left(\mathbf{x}_i - \frac{m}{n}\boldsymbol{\nu}\right)^T\mathbf{y} \right] + s. \quad (55)$$

Now, (55) can be compactly represented by

$$\min. \quad a\mathbf{y}^T\mathbf{y} + 2\mathbf{b}^T\mathbf{y} + c \quad (56)$$

where

$$a = w_q(n-m) + w_\mu \frac{m}{n}(n-m) \quad (57)$$

$$\mathbf{b} = -w_\mu(n-m)\frac{m}{n}\boldsymbol{\nu} \quad (58)$$

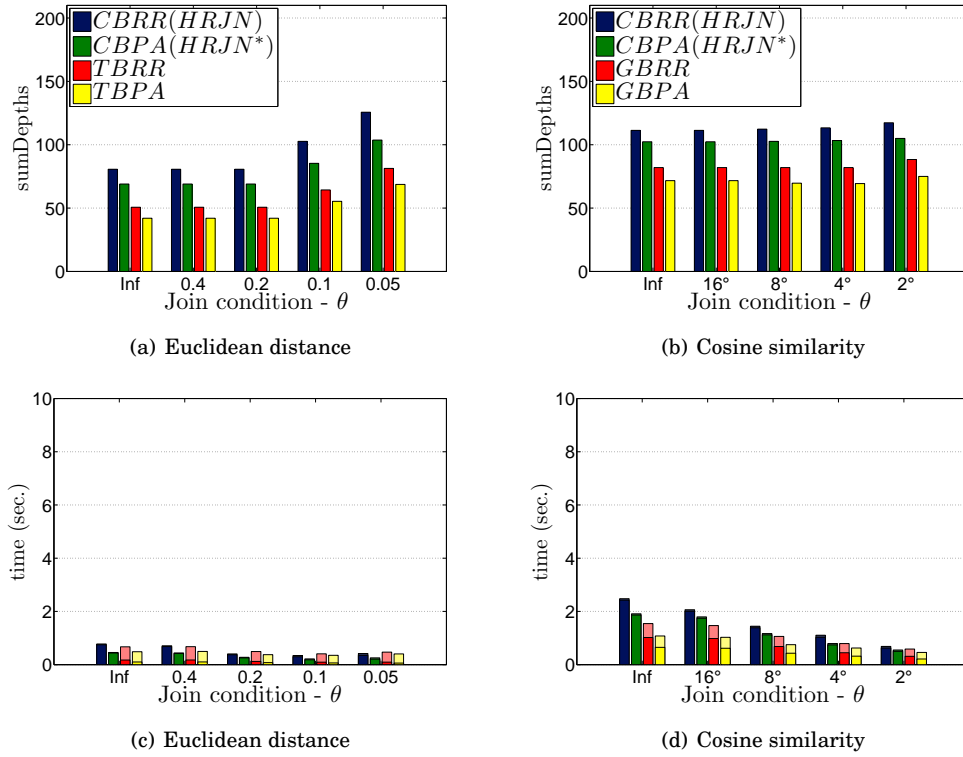


Fig. 20. Performance of the tested algorithms as a function of the join condition θ for *distance-based access*.

$$c = w_{\mu}(n - m) \frac{m^2}{n^2} \boldsymbol{\nu}^T \boldsymbol{\nu} + w_{\mu} \sum_{i=1}^m \left(\mathbf{x}_i - \frac{m}{n} \boldsymbol{\nu} \right)^T \left(\mathbf{x}_i - \frac{m}{n} \boldsymbol{\nu} \right) + s \quad (59)$$

The solution of the problem

$$\begin{aligned} & \text{minimize } a\mathbf{y}^T \mathbf{y} + 2\mathbf{b}^T \mathbf{y} + c \\ & \text{subject to } \|\mathbf{y}\| \geq \delta \end{aligned} \quad (60)$$

is obtained by imposing the Karush-Kuhn-Tucker conditions and it is given by

$$\mathbf{y}^* = \begin{cases} -\mathbf{b}/a & \text{if } \|\mathbf{b}/a\| \geq \delta \\ \delta \frac{\boldsymbol{\nu}}{\|\boldsymbol{\nu}\|} & \text{otherwise} \end{cases} \quad (61)$$

or, equivalently,

$$\mathbf{y}^* = \begin{cases} \boldsymbol{\nu} \frac{mw_{\mu}}{mw_{\mu} + nw_q} & \text{if } \left\| \boldsymbol{\nu} \frac{mw_{\mu}}{mw_{\mu} + nw_q} \right\| \geq \delta \\ \boldsymbol{\nu} \frac{\delta}{\|\boldsymbol{\nu}\|} & \text{otherwise} \end{cases} \quad (62)$$

Note that \mathbf{y}^* has the same direction as the vector representing the centroid $\boldsymbol{\nu}$. If $\mathbf{q} \neq \mathbf{0}$, then $\mathbf{y}^* \leftarrow \mathbf{q} + \mathbf{y}^*$, thus:

$$\mathbf{y}^* = \begin{cases} \mathbf{q} + (\boldsymbol{\nu} - \mathbf{q}) \frac{mw_{\mu}}{mw_{\mu} + nw_q} & \text{if } \left\| (\boldsymbol{\nu} - \mathbf{q}) \frac{mw_{\mu}}{mw_{\mu} + nw_q} \right\| \geq \delta \\ \mathbf{q} + (\boldsymbol{\nu} - \mathbf{q}) \frac{\delta}{\|\boldsymbol{\nu} - \mathbf{q}\|} & \text{otherwise} \end{cases} \quad (63)$$

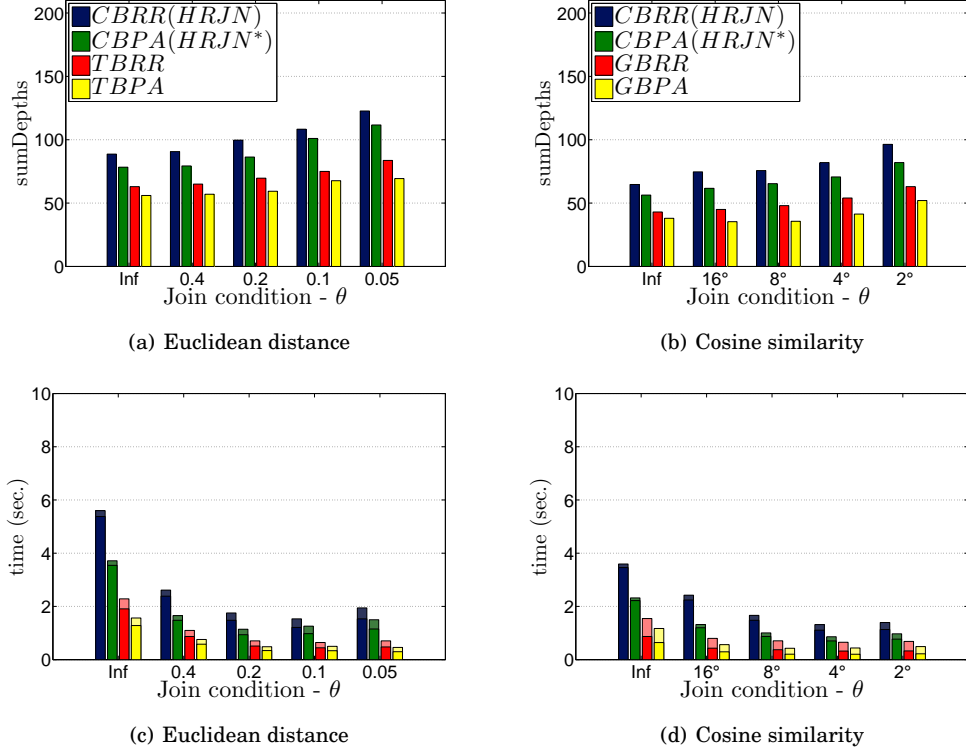


Fig. 21. Performance of the tested algorithms as a function of the join condition θ for *score-based access*.

A.2. Derivation of (19)

The definition of $h(\mathbf{y}_{m+1}, \dots, \mathbf{y}_n)$ given in (18) can be expanded as follows.

$$\begin{aligned}
 & h(\mathbf{y}_{m+1}, \dots, \mathbf{y}_n) \\
 &= \sum_{i=1}^m \|\mathbf{x}_i - \boldsymbol{\mu}\|^2 + \sum_{i=m+1}^n \|\mathbf{y}_i - \boldsymbol{\mu}\|^2 \\
 &= \sum_{i=1}^m \left\| \mathbf{x}_i - \frac{1}{n} \left(\sum_{j=1}^m \mathbf{x}_j + \sum_{j=m+1}^n \mathbf{y}_j \right) \right\|^2 + \sum_{i=m+1}^n \left\| \mathbf{y}_i - \frac{1}{n} \left(\sum_{j=1}^m \mathbf{x}_j + \sum_{j=m+1}^n \mathbf{y}_j \right) \right\|^2 \\
 &= \sum_{i=1}^m \left[\mathbf{x}_i^T \mathbf{x}_i + \frac{1}{n^2} \left(\sum_{j=1}^m \mathbf{x}_j + \sum_{j=m+1}^n \mathbf{y}_j \right)^T \left(\sum_{j=1}^m \mathbf{x}_j + \sum_{j=m+1}^n \mathbf{y}_j \right) - \frac{2}{n} \left(\sum_{j=1}^m \mathbf{x}_j + \sum_{j=m+1}^n \mathbf{y}_j \right)^T \mathbf{x}_i \right] + \\
 & \quad \sum_{i=m+1}^n \left[\mathbf{y}_i^T \mathbf{y}_i + \frac{1}{n^2} \left(\sum_{j=1}^m \mathbf{x}_j + \sum_{j=m+1}^n \mathbf{y}_j \right)^T \left(\sum_{j=1}^m \mathbf{x}_j + \sum_{j=m+1}^n \mathbf{y}_j \right) - \frac{2}{n} \left(\sum_{j=1}^m \mathbf{x}_j + \sum_{j=m+1}^n \mathbf{y}_j \right)^T \mathbf{y}_i \right] \\
 &= \sum_{i=1}^m \mathbf{x}_i^T \mathbf{x}_i + \frac{m}{n^2} \left(\sum_{j=1}^m \mathbf{x}_j + \sum_{j=m+1}^n \mathbf{y}_j \right)^T \left(\sum_{j=1}^m \mathbf{x}_j + \sum_{j=m+1}^n \mathbf{y}_j \right) - \frac{2}{n} \left(\sum_{j=1}^m \mathbf{x}_j + \sum_{j=m+1}^n \mathbf{y}_j \right)^T \sum_{i=1}^m \mathbf{x}_i + \\
 & \quad \sum_{i=m+1}^n \mathbf{y}_i^T \mathbf{y}_i + \frac{n-m}{n^2} \left(\sum_{j=1}^m \mathbf{x}_j + \sum_{j=m+1}^n \mathbf{y}_j \right)^T \left(\sum_{j=1}^m \mathbf{x}_j + \sum_{j=m+1}^n \mathbf{y}_j \right) - \frac{2}{n} \left(\sum_{j=1}^m \mathbf{x}_j + \sum_{j=m+1}^n \mathbf{y}_j \right)^T \sum_{i=m+1}^n \mathbf{y}_i
 \end{aligned}$$

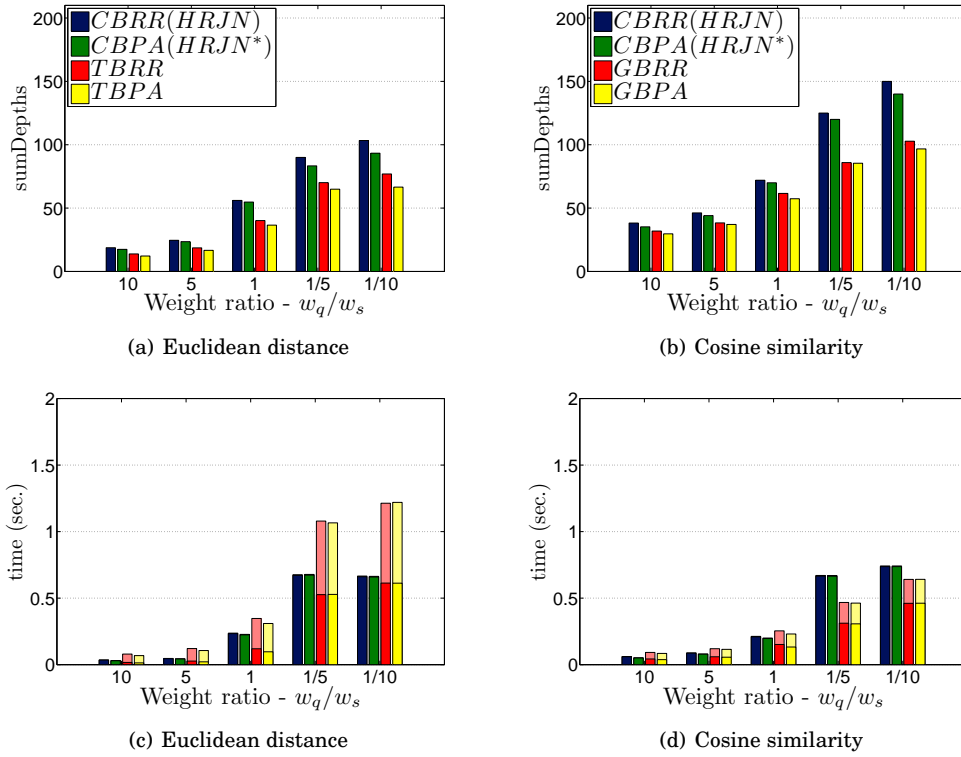


Fig. 22. Performance of the tested algorithms as a function of the weight ratio w_q/w_s for distance-based access.

$$\begin{aligned}
&= \sum_{i=1}^m \mathbf{x}_i^T \mathbf{x}_i + \sum_{i=m+1}^n \mathbf{y}_i^T \mathbf{y}_i - \frac{1}{n} \left(\sum_{j=1}^m \mathbf{x}_j + \sum_{j=m+1}^n \mathbf{y}_j \right)^T \left(\sum_{j=1}^m \mathbf{x}_j + \sum_{j=m+1}^n \mathbf{y}_j \right) \\
&= \sum_{i=1}^m \mathbf{x}_i^T \mathbf{x}_i + \sum_{i=m+1}^n \mathbf{y}_i^T \mathbf{y}_i - \frac{1}{n} \left(\sum_{j=1}^m \mathbf{x}_j \right)^T \left(\sum_{j=1}^m \mathbf{x}_j \right) - \frac{1}{n} \left(\sum_{j=m+1}^n \mathbf{y}_j \right)^T \left(\sum_{j=m+1}^n \mathbf{y}_j \right) - \frac{2}{n} \left(\sum_{j=1}^m \mathbf{x}_j \right)^T \left(\sum_{j=m+1}^n \mathbf{y}_j \right) \\
&= \sum_{i=1}^m \mathbf{x}_i^T \mathbf{x}_i + \sum_{i=m+1}^n \mathbf{y}_i^T \mathbf{y}_i - \frac{m^2}{n} \boldsymbol{\nu}^T \boldsymbol{\nu} - \frac{1}{n} \left(\sum_{j=m+1}^n \mathbf{y}_j \right)^T \left(\sum_{j=m+1}^n \mathbf{y}_j \right) - 2 \frac{m}{n} \boldsymbol{\nu}^T \left(\sum_{j=m+1}^n \mathbf{y}_j \right)
\end{aligned}$$

The last line equals the right-hand side of (19).

A.3. Solution of (24)

Let $\boldsymbol{\theta} = [\theta_1, \theta_2, \dots, \theta_n]^T$ denote a vector representing the distances from \mathbf{q} of the n vectors of the tuples that form a combination. Problem (24) can be written as a quadratic program (QP) in canonical form, as follows

$$\begin{aligned}
&\text{minimize } \boldsymbol{\theta}^T \mathbf{H} \boldsymbol{\theta} \\
&\text{subject to } \mathbf{A}_{e\mathbf{q}} \boldsymbol{\theta} = \mathbf{b}_{e\mathbf{q}} \\
&\quad \boldsymbol{\theta} \geq \boldsymbol{\ell}
\end{aligned} \tag{64}$$

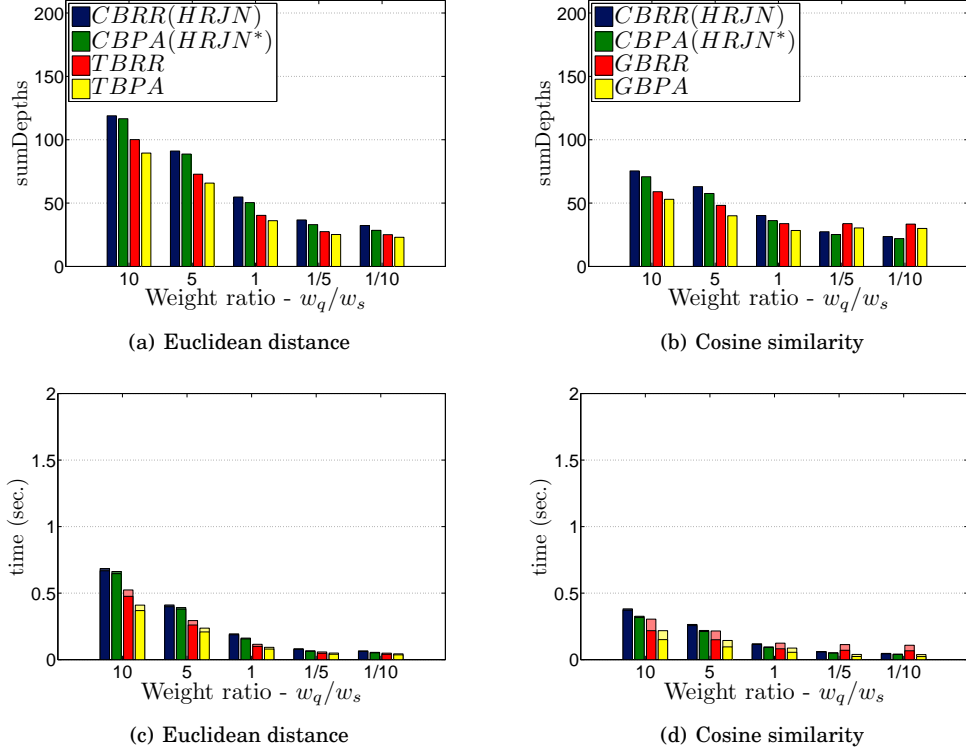


Fig. 23. Performance of the tested algorithms as a function of the weight ratio w_q/w_s for *score-based access*.

To see this, let $\mathbf{1} = [1, 1, \dots, 1]^T \in \mathbb{R}^n$ and \mathbf{I}_n be the $n \times n$ identity matrix. The objective function of (24) can be written in matrix form:

$$\begin{aligned}
 & \sum_{i=1}^m w_s \ln(\sigma(\tau_i)) + \sum_{i=m+1}^n w_s \ln(\sigma_i^{\max}) - \sum_{i=1}^n w_q \theta_i^2 - \sum_{i=1}^n w_\mu \left(\theta_i - \frac{1}{n} \sum_{j=1}^n \theta_j \right)^2 = \\
 & = r - w_q \boldsymbol{\theta}^T \boldsymbol{\theta} - w_\mu \left(\boldsymbol{\theta} - \frac{1}{n} \mathbf{1} \cdot \mathbf{1}^T \boldsymbol{\theta} \right)^T \left(\boldsymbol{\theta} - \frac{1}{n} \mathbf{1} \cdot \mathbf{1}^T \boldsymbol{\theta} \right) \\
 & = r - w_q \boldsymbol{\theta}^T \boldsymbol{\theta} - w_\mu \boldsymbol{\theta}^T \left(\mathbf{I}_n - \frac{1}{n} \mathbf{1} \cdot \mathbf{1}^T \right)^T \left(\mathbf{I}_n - \frac{1}{n} \mathbf{1} \cdot \mathbf{1}^T \right) \boldsymbol{\theta}
 \end{aligned}$$

where r is an expression not containing $\boldsymbol{\theta}$. Therefore:

$$\mathbf{H} = w_q \mathbf{I} + w_\mu \left(\mathbf{I}_n - \frac{1}{n} \mathbf{1} \cdot \mathbf{1}^T \right)^T \left(\mathbf{I}_n - \frac{1}{n} \mathbf{1} \cdot \mathbf{1}^T \right) \quad (65)$$

From the equality constraints in (24), $\theta_i = \mathcal{P}(\mathbf{x}(\tau_i)), i = 1, \dots, m$, it follows

$$\mathbf{A}_{eq} = \begin{bmatrix} \mathbf{I}_m & \mathbf{0}_{m \times (n-m)} \\ \mathbf{0}_{(n-m) \times m} & \mathbf{0}_{(n-m) \times (n-m)} \end{bmatrix} \quad (66)$$

$$\mathbf{b}_{eq} = [\mathcal{P}(\mathbf{x}(\tau_1)), \dots, \mathcal{P}(\mathbf{x}(\tau_m)), \mathbf{0}_{1 \times (n-m)}]^T \quad (67)$$

Finally, from the inequality constraints in (24), $\theta_i \geq \delta_i, i = m + 1, \dots, n$, it follows

$$\boldsymbol{\ell} = [\mathbf{0}_{m \times 1}, \delta_{m+1}, \dots, \delta_n]^T \quad (68)$$

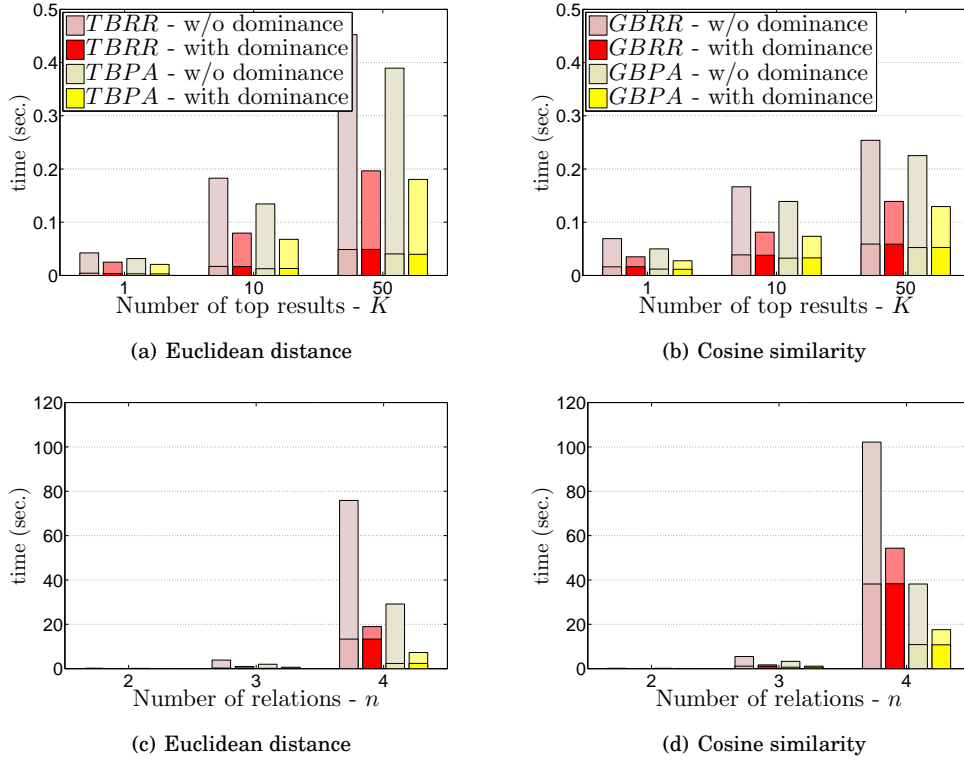


Fig. 24. Impact of dominance on the tested algorithms in terms of the *total CPU time* metrics.

A.4. Solution of (35)

We want to solve the following (nonconvex) quadratically constrained quadratic problem (QCQP).

$$\begin{aligned}
 \max. \quad & w_q \sum_{i=1}^n \tilde{\mathbf{q}}^T \tilde{\mathbf{y}}_i + \frac{w_\mu}{\|\tilde{\boldsymbol{\mu}}\|} \sum_{i=1}^n \tilde{\boldsymbol{\mu}}^T \tilde{\mathbf{y}}_i + c \\
 \text{s.t.} \quad & \tilde{\mathbf{y}}_i = \tilde{\mathbf{x}}_i, i \in \{1, \dots, m\} \\
 & \tilde{\mathbf{q}}^T \tilde{\mathbf{y}}_i \leq \delta_i, i \in \{m+1, \dots, n\} \\
 & \tilde{\mathbf{y}}_i^T \tilde{\mathbf{y}}_i = 1, i \in \{1, \dots, n\}
 \end{aligned} \tag{69}$$

First, let us consider the following (simpler) problem

$$\begin{aligned}
 \max. \quad & w_q \sum_{i=1}^r \tilde{\mathbf{q}}^T \tilde{\mathbf{z}}_i + w_q \sum_{i=r+1}^n \tilde{\mathbf{q}}^T \tilde{\mathbf{y}}_i + \frac{w_\mu}{\|\tilde{\boldsymbol{\mu}}\|} \sum_{i=1}^r \tilde{\boldsymbol{\mu}}^T \tilde{\mathbf{z}}_i + \frac{w_\mu}{\|\tilde{\boldsymbol{\mu}}\|} \sum_{i=r+1}^n \tilde{\boldsymbol{\mu}}^T \tilde{\mathbf{y}}_i + c \\
 \text{s.t.} \quad & \tilde{\mathbf{y}}_i^T \tilde{\mathbf{y}}_i = 1, i \in \{r+1, \dots, n\}
 \end{aligned} \tag{70}$$

where $r \geq m$ and $\tilde{\mathbf{z}}_i$ are constant vectors, i.e. $\tilde{\mathbf{z}}_i = \tilde{\mathbf{x}}_i, i \in \{1, \dots, m\}$, $\tilde{\mathbf{z}}_i = \tilde{\mathbf{y}}_i, i \in \{m+1, \dots, r\}$. We recognize that (70) is symmetric with respect to the unknown variables $\tilde{\mathbf{y}}_i, i \in \{r+1, \dots, n\}$. Hence, its optimal solution is such that $\tilde{\mathbf{y}}_{r+1}^* = \dots = \tilde{\mathbf{y}}_n^* = \tilde{\mathbf{y}}^*$. Hence, we can reduce (70) into a problem in a single unknown vector $\tilde{\mathbf{y}}$:

$$\begin{aligned}
 \max. \quad & \mathbf{a}^T \tilde{\mathbf{y}} + b \\
 \text{s.t.} \quad & \tilde{\mathbf{y}}^T \tilde{\mathbf{y}} = 1
 \end{aligned} \tag{71}$$

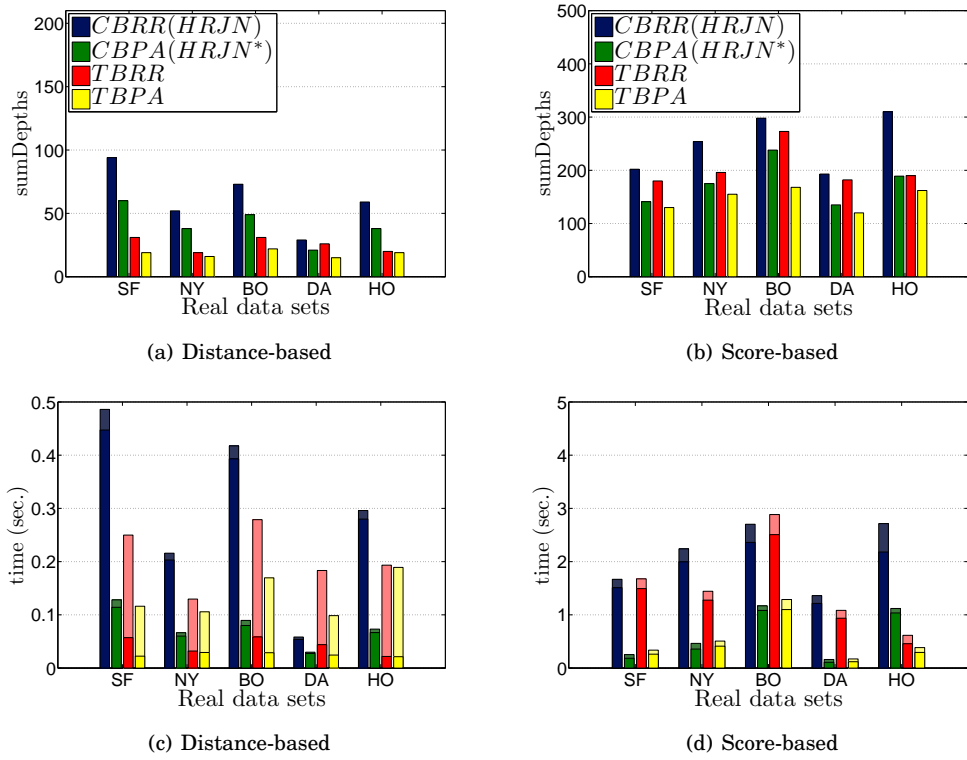


Fig. 25. Performance of the tested algorithms for real data sets in terms of both the *sumDepths* and the *total CPU time* metrics with Euclidean distance for distance-based and score-based access.

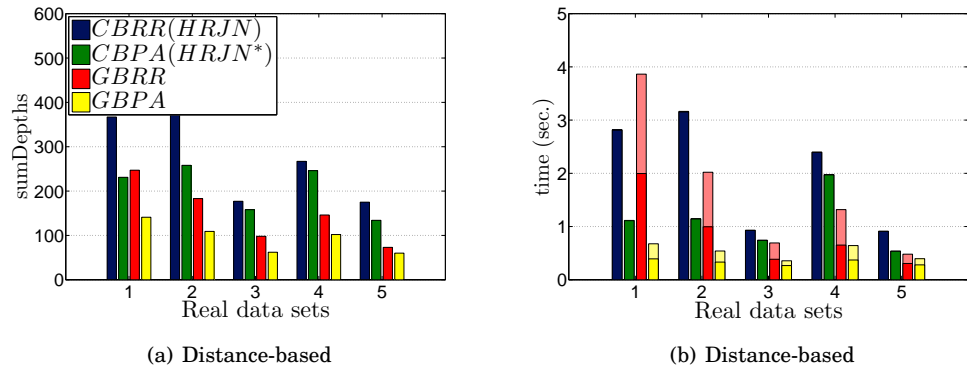


Fig. 26. Performance of the tested algorithms for real data sets in terms of both the *sumDepths* and the *total CPU time* metrics with cosine similarity for distance-based access.

where

$$\mathbf{a} = w_q(n-r)\tilde{\mathbf{q}} + 2\frac{w_\mu}{\|\tilde{\boldsymbol{\mu}}\|}\frac{n-r}{n}r\tilde{\boldsymbol{\nu}} \quad (72)$$

$$b = w_q r \tilde{\mathbf{q}}^T \tilde{\boldsymbol{\nu}} + \frac{w_\mu}{\|\tilde{\boldsymbol{\mu}}\|}\frac{(n-r)^2}{n} + \frac{w_\mu}{\|\tilde{\boldsymbol{\mu}}\|}\frac{r^2}{n}\tilde{\boldsymbol{\nu}}^T \tilde{\boldsymbol{\nu}} + c \quad (73)$$

and $\tilde{\boldsymbol{\nu}} = \frac{1}{r} \sum_{i=1}^r \tilde{\mathbf{z}}_i$. The optimal solution of (71) is

$$\tilde{\mathbf{y}}^* = \frac{\mathbf{a}}{\|\mathbf{a}\|} \quad (74)$$

The solution of (35) can be found by writing the Karush-Kuhn-Tucker conditions. Given the complementary slackness conditions

$$\lambda_i(\tilde{\mathbf{q}}^T \tilde{\mathbf{y}} - \delta_i) = 0, \quad i \in \{m+1, \dots, n\} \quad (75)$$

each of the $n-m$ Lagrange multipliers λ_i can either be equal to zero or positive. W.l.o.g. consider one of the 2^{n-m} possible cases, corresponding to $\lambda_i > 0$, $i = \{m+1, \dots, r\}$ and $\lambda_i = 0$, $i = \{r+1, \dots, n\}$. Due to complementary slackness, this implies $\tilde{\mathbf{q}}^T \tilde{\mathbf{y}}^* = \delta_i$, $i = \{m+1, \dots, r\}$. This scenario leads to solving problem (70). If the optimal solution of (70) is feasible for the original problem (35), i.e. $\tilde{\mathbf{q}}^T \tilde{\mathbf{y}}^* \leq \delta_i$, $i = \{r+1, \dots, n\}$, then it is marked as a candidate solution. By enumerating the 2^{n-m} cases it is possible to find up to 2^{n-m} candidate solutions. The optimal solution of (35) is the candidate solution that maximizes the objective function of (35).

A.5. Solution of (49)

We adopt the shorthand notation $\mathbf{x}_i = \mathbf{x}(\tau_i)$. The centroid $\boldsymbol{\mu}$ of the full combination can be written as

$$\boldsymbol{\mu} = \frac{1}{n} \left(\sum_{i=1}^m \mathbf{x}_i + (n-m)\mathbf{y} \right) = \frac{1}{n} \sum_{i=1}^m \mathbf{x}_i + \frac{n-m}{n}\mathbf{y} = \frac{m}{n}\boldsymbol{\nu} + \frac{n-m}{n}\mathbf{y} \quad (76)$$

where $\boldsymbol{\nu} = \frac{1}{m} \sum_{i=1}^m \mathbf{x}_i$ is the centroid of the partial combination.

The objective function (49) can be rewritten as follows:

$$\begin{aligned} \max. \quad & (n-m) \left[w_q \mathbf{q} + 2 \frac{w_\mu}{\|\tilde{\boldsymbol{\mu}}\|} \frac{m}{n} \boldsymbol{\nu} \right]^T \mathbf{y} + \sum_{i=1}^m w_s \sigma(\tau_i) + w_q \mathbf{q}^T \mathbf{x}(\tau_i) + \sum_{i=m+1}^n w_s \sigma(R[p_i]) + \\ & \frac{w_\mu}{\|\tilde{\boldsymbol{\mu}}\|} \frac{m^2}{n} \boldsymbol{\nu}^T \boldsymbol{\nu} + \frac{w_\mu}{\|\tilde{\boldsymbol{\mu}}\|} \frac{(n-m)^2}{n}, \quad \text{i.e.,} \\ \max. \quad & \mathbf{b}^T \mathbf{y} + c \\ \text{s.t.} \quad & \mathbf{y}^T \mathbf{y} = 1 \end{aligned} \quad (77)$$

where

$$\mathbf{b} = (n-m) \left[w_q \mathbf{q} + 2 \frac{w_\mu}{\|\tilde{\boldsymbol{\mu}}\|} \frac{m}{n} \boldsymbol{\nu} \right] \quad (78)$$

and c is a constant that does not depend on \mathbf{y} . The solution of the problem is obtained by imposing the Karush-Kuhn-Tucker conditions and it is given by

$$\mathbf{y}^* = \frac{\mathbf{b}}{\sqrt{\mathbf{b}^T \mathbf{b}}} \quad (79)$$