FALCON: Feedback Adaptive Loop for Content-based Retrieval

Leejay Wu Christos Faloutsos Katia Sycara
Terry R. Payne

June 2000
CMU-CS-00-142

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

This research was partially funded by the National Science Foundation under Grants No. IRI-9625428, DMS-9873442, and IIS-9910606; also, by the National Science Foundation, ARPA and NASA under NSF Cooperative Agreement No. IRI-9411299; DARPA/ITO through Order F463, issued by ESC/ENS under contract N66001-97-C-851; and the Office of Naval Research Grant N-00014-96-1-1222. Additional funding was provided by donations from NEC and Intel.

Keywords: Databases, Information Retrieval, Multimedia

Abstract

Several methods currently exist that can perform relatively simple queries driven by relevance feedback on large multimedia databases. However, all these methods work only for vector spaces; that is, they require that objects be represented as vectors within feature spaces. Moreover, their implied query regions are typically convex. This research paper explains our solution.

We propose a novel method that is designed to handle disjunctive queries within metric spaces. The user provides weights for positive examples; our system "learns" the implied concept and returns similar objects. Our method differs from existing relevance-feedback methods that base themselves upon Euclidean or Mahalanobis metrics, as it facilitates learning even disjunctive, concave models within vector spaces, as well as arbitrary metric spaces.

Our main contributions are two-fold. Not only do we present a novel way to estimate the dissimilarity of an object to a set of desirable objects, but we support it with an algorithm that shows how to exploit metric indexing structures that support range queries to accelerate the search without incurring false dismissals. Our empirical results demonstrate that our method converges rapidly to excellent precision/recall, while outperforming sequential scanning by up to 200%.

1 Introduction

As the size and diversity of multimedia databases increases, there is a growing need for systems that can process complex queries which may be be disjunctive or otherwise correspond to arbitrarily shaped regions when mapped into some query space. Such queries may have underlying concepts that cannot be easily expressed. For example, there are several methods that can generate single hyper-ellipsoids within some feature space for simple queries, but fail to identify appropriate regions for disjoint or concave queries. In addition, some mechanism is needed to assist database users in defining or expressing their requirements. This paper presents a novel approach, *FALCON*, which allows easy specification of complex queries, within both vector and metric spaces, for multimedia and traditional databases.

Many retrieval methods rely on representing database entries as points within some feature space, where the topology of this space is dependent on the characteristics of the database, such as the indices used and the types of values in those indices. Often, they rely on the assumption that adjacent points within this space map to very similar database entries [9]. Distance functions are chosen to score dissimilarity between points within this space. Thus, one may query a database by performing either a range query or a nearest-neighbor search relative to a single point or hyper-surface within this space.

However, users may wish to perform more complicated queries. On a real-world database, a user may wish to retrieve a class of objects that does not map to a contiguous region according to the similarity metric. In addition, there exist metric databases in which there are no explicit features, but instead just a distance function. These functions are called distance metrics, provided that they satisfy symmetry and triangle inequality criteria [4], resulting in metric spaces. Finding similar objects within this space is related to clustering, which can be done in metric spaces as per [15].

This can be illustrated by means of an example. Images may form a metric space, based upon a variety of *image-image* distance metrics. If one metric is heavily dependent on colors instead of other features such as shapes, then problems may arise when indexing a database of bird photographs. If one searches for a bird that varies in coloration (such as a parrot), then it can be very difficult to retrieve suitable images. Other multimedia databases, such as the Informedia Project[1] archive digital video. Queries that are easily expressed in natural language, such as *"retrieve all video footage which feature heads of state"* may prove to be far too diverse to be adequately represented by a single distance metric laboring under the restriction of a single cluster.

FALCON attempts to overcome these problems by combining distances and incorporating

[1] `http://informedia.cs.cmu.edu`

user feedback in such a way as to "learn" the nature of such queries. One important feature of this combination model – the *aggregate dissimilarity* model – is that it can be applied to metric data sets, as it does not use any information about the data itself aside from that returned by a pairwise distance metric. The mechanism used to achieve this is described below and experiments on synthetic and real data sets are presented. These experiments demonstrate that FALCON can provide high-quality results in terms of precision and recall after 5 to 20 iterations.

The paper is organized as follows: Section 2 summarizes a sample of existing related systems; the mechanisms underlying FALCON are presented in Section 3; the experiments and corresponding results are detailed in Section 4, and the subsequent analysis is presented in Section 5. The paper concludes with Section 6.

2 Related Work

A number of systems have been developed that attempt to determine the user's requirements from a set of examples. These systems are designed to work with the general class of problems that involve example-based queries. Note they handle *neither* unusual disjoint queries, nor arbitrary metric spaces:

- Rocchio's relevance feedback mechanism [11], which generates hyper-spherical isosurfaces in feature space.
- MARS (Multimedia Analysis and Retrieval System) [10, 12, 13], which includes a query expansion model that can weight features from multiple objects. However, MARS limits the sums of weights, making it difficult to specify that being similar to any single object from a set might be sufficient for a very good score. This means that arbitrarily disjunctive queries are effectively impossible to support.
- MindReader, which uses the Mahalanobis distance to allow arbitrarily oriented ellipsoids [6]. The Mahalanobis distance $M(\vec{x}, \vec{y})$ is defined as $(\vec{x} - \vec{y})^T \times \mathbf{M} \times (\vec{x} - \vec{y})$ where $\vec{x}$ and $\vec{y}$ are n-dimensional column vectors and $\mathbf{M}$ is a $n \times n$ matrix. This corresponds to a weighted Euclidean distance, and permits effective rotation of the axes, but requires many examples to calculate the covariance matrix.

The original assumption behind the earliest systems was that the user has an ideal point in vector space that he is looking for; one can then try to determine both the ideal point and the relative weights of the axes. It is exactly this dependence on a vector representation of data that *prevents* all past methods from generalizing to metric spaces.

Later systems, such as current versions of MARS, have shifted towards query expansion models which permit not just query point movement but also actual elaboration by weighting the relative importance of different features of multiple positive examples [10]. However, one should note that MARS has been specialized for image databases with features, whereas MindReader generates isosurfaces consisting of single hyper-ellipsoids, and therefore does not handle disjunctive queries.

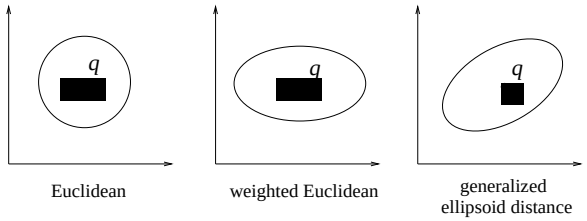


Figure 1: Generic isosurfaces for three previous methods.

See Figure 1 for an illustration of these types of isosurfaces.

Fox and Salton used the L_p metric to surpass fuzzy Boolean methods in the domain of text retrieval, replacing the use of minimum and maximum functions. Our objective is similar, but more complex; we wish to model arbitrarily disjunctive example-based queries in unbounded metric spaces using relevance feedback but lacking specific features [14].

What we do differs from these previous methods. We do not rely on vector spaces; require user input beyond that of relevance feedback of examples; or sacrifice the ability to use disjunctive queries that correspond to arbitrary groupings in metric spaces.

In addition, we show that advanced indexing methods can be used to significantly speed up the search process. Spatial indexing methods such as R-trees [5] and R*-trees [1, 2] would serve if we were to limit ourselves to vector domains; M-trees [3] provide fast range-queries in general metric spaces.

3 Proposed Method

This section proposes the underlying mechanisms and assumptions made by FALCON. Table 1 lists the notation used in this section.

Let $\mathcal{X}$ be the set of objects in our metric data set. Then, let $d : \mathcal{X} \times \mathcal{X} \rightarrow \Re$ be the provided distance metric that defines our metric space.

To start each query, the user specifies at least one "desirable" example that is representative of the intended query. This set of "good" examples is denoted by $\mathcal{G}$. Thus:

Symbol	Description
$\mathcal{X}$	The set of objects in our data set.
x	Any single object from $\mathcal{X}$.
$\mathcal{G}$	The current set of user-specified "good points".
g_i	A member of $\mathcal{G}$.
$D_{\mathcal{G}}$	The aggregate dissimilarity function based on $\mathcal{G}$.
$D_{\mathcal{G}}(x)$	The aggregate dissimilarity value for object x to the current good set $\mathcal{G}$.
α	A constant that influences how $D_{\mathcal{G}}$ behaves.
d	The pairwise dissimilarity function.

Table 1: Notation used within this paper.

Problem 1: Query by Multiple Examples	
Given	$\mathcal{X}$, a metric data set
	d, its pairwise distance metric
	$\mathcal{G} = \{g_i\}$, the set of "good objects"
Find	Other desirable objects in $\mathcal{X}$ that are similar to $\mathcal{G}$.

We propose solving this by determining a scoring function that models the user's query. Namely, we seek a function $D_{\mathcal{G}} : \mathcal{X} \rightarrow \Re$ based on $\mathcal{G}$, such that this function varies inversely with the desirability of x.

3.1 Proposed "Aggregate Dissimilarity" Function

Once we find a function that fulfills our requirements – namely, ranking objects inversely according to their apparent desirability as compared to $\mathcal{G}$ – the problem of finding relevant objects reduces to that of sorting. We define such a function as follows:

Problem 2: Aggregate Dissimilarity	
Given	$x \in \mathcal{X}$, a candidate
	$\mathcal{G} = \{g_i\}$, the set of user-selected "good objects"
	d, pairwise distance metric for $\mathcal{X}$
Determine	$D_{\mathcal{G}}(x)$, its aggregate dissimilarity

We propose that the FALCON aggregate dissimilarity, $D_{\mathcal{G}}(x)$ be computed as the α^{th} root of the arithmetic mean of the α^{th} powers, of the pairwise distances, as expressed by

$$(D_{\mathcal{G}}(x))^{\alpha} = \begin{cases} 0 & '' \quad (\alpha < 0) \bigwedge \exists i \, d(x, g_i) = 0 \\ \frac{1}{k} \times \sum_{i=1}^{k} d(x, g_i)^{\alpha} & '' \quad \text{otherwise} \end{cases} \tag{1}$$

If the value of α is very high, the highest distance will have the largest impact on $D_{\mathcal{G}}(x)$, while the reverse is true for very low values of α. This method is justified in the Appendix A.

Note that Equation 1 has the following properties:

- As the user modifies the set $\mathcal{G}$ of desirable objects, the function changes to reflect this. Current incarnations do not automatically add or remove objects; one reasonable approach might be to remove from $\mathcal{G}$ objects that are very similar to additions, in order to reduce redundancy. We do not prune outliers, since they may be necessary to support unusual queries.

- Negative values of α mimic a fuzzy OR function; positive values mimic an AND function. See the Appendix for details.

- For $\alpha \leq 0$, d such that $\forall x \, d(x, x) = 0$ and $x \in \mathcal{G}$, then $D_{\mathcal{G}}(x) = 0$ – in other words, with a fuzzy OR, exactly matching a single $x \in \mathcal{G}$ yields the lowest possible score – even if we generalize $D_{\mathcal{G}}$ to accept weights, as described later. With $\alpha > 0$, FALCON's behavior is closer to that of MARS in that objects need to be similar to everything in $\mathcal{G}$ to minimize computed distance [10].

- $\alpha = -1$ is the harmonic mean of the pairwise distances.

- $\alpha = 1$ is the arithmetic mean of the pairwise distances.

- One can show using L'Hopital's rule,that as α approaches 0, $DG(x)$ approaches the geometric mean of the pairwise distances. The proof, however, is too long to include.

Since it is not obvious which values of α are the most suitable, we empirically compare results for various values of α. Our upcoming experiments show that $\alpha = -5$ is a reasonable choice.

With regards to the previous example regarding color spaces, photos, and parrots, one possible sequence of events would be the following.

1. The user chooses a data set, $\mathcal{X}$, consisting of bird photos. This data set is paired with a pairwise distance metric d, which relies primarily on coloration when comparing images.

2. The user first chooses the image of a popular green-and-red parrot. This becomes the first member of $\mathcal{G}$.

3. FALCON returns the best matches, according to $D_{\mathcal{G}}$; note that grey parrots might be ranked below cardinals – which are mostly red – and peacocks – which tend to be green.

4. The user adds a picture of a grey parrot to the good set, which alters the aggregate dissimilarity function $D_{\mathcal{G}}$. Other images of grey parrots will now be considered more relevant to the query.

5. Repeat as desired; depending upon the data and the metric, convergence may be fairly rapid.

Figure 2 illustrates this sequence with a stocks database.

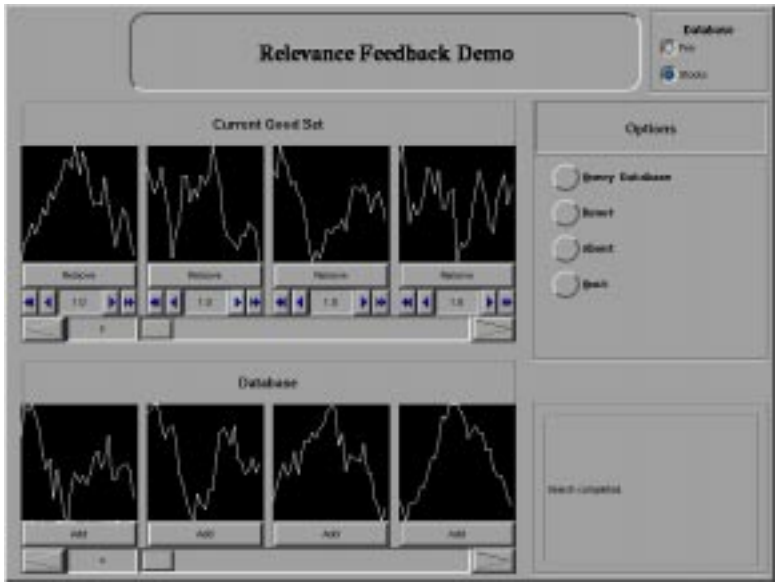


Figure 2: This shows our prototype working on the STOCKS data set. The top four stocks are the chosen examples; all four have either a sharp rise or a sharp fall in the middle before returning to near the original price. The bottom four are among those that ranked highest according to the query. The various buttons allow for choosing the "good" set as well weighting each member of that set. It must be emphasized that this query is disjunctive, and that other methods would face severe difficulties accordingly.

One useful generalization of the FALCON distance is to allow for weights. Unlike a distance "combination" function that only uses the minimum distance, the FALCON distance is easily

modified to accept a w_i term for numerical feedback from a user. The proposed extension takes the form:

$$(D_{\mathcal{G}}(x))^{\alpha} = \frac{1}{\sum_{i=1}^{k} w_i} \cdot \sum_{i=1}^{k} w_i (d(x, g_i))^{\alpha} \tag{2}$$

This has the effect that distance to the favored objects is penalized less if α is negative. We do *not* raise the weights to the α^{th} power, as we believe that this – which would have the opposite effect – would be much less intuitive. In addition, we retain the influence of all pairwise distances between candidate and examples in all cases except that of an exact match, unlike a pure minimization function.

3.2 Speed and Completeness

There is the obvious question of speed. Sequential scanning would entail computing aggregate dissimilarity via Equation 1 or 2 for every element of the database, which would require $\Theta(nk)$ in the general case where we have n objects in $\mathcal{X}$ and k in $\mathcal{G}$. Our objective is to return the same results as a sequential scan, with less work on average per search.

Index structures which support fast range queries can be used to achieve significant speed-ups. First, we shift focus to the aggregate dissimilarity version of the range query:

Problem 3: Range Query by Multiple Example	
Given	$\mathcal{X}$, the database
	$\mathcal{G} = \{g_i\}$, the set of "good objects"
	ϵ, a threshold
	d, pairwise distance metric for $\mathcal{X}$
Find	Q, such that $Q = \{x : x \in \mathcal{X} \wedge D_{\mathcal{G}}(x) < \epsilon\}$ quickly

Theorem 1 *Consider Problem 3 above. Executing $k = |\mathcal{G}|$ separate range queries, each with the same threshold, ϵ, will yield k sets, the union of which forms a superset of the actual answer. No object will be falsely discarded as a candidate by such a procedure.*

To prove this, we use the following lemma.

Lemma 1 *Suppose a given object x is not in any of the lists. Then, for every good object $\}_i$, $d(x, \}_i) > \epsilon$.*

Proof: Let us divide this proof into two cases, depending upon the sign of α.

- Suppose α is negative. Then:

$$\forall i \; d(x, \mathcal{G}_i)^\alpha < \epsilon^\alpha. \tag{3}$$

Multiply by the weights (which we require to be positive), if any, and sum to yield the following.

$$\sum_{i=1}^{k} (w_i \times d(x, \mathcal{G}_i)^\alpha) < \sum_{i}^{k} w_i \epsilon^\alpha \tag{4}$$

Divide both sides by the sum of the weights, to get

$$\frac{\sum_i^k w_i \times d(x, \mathcal{G}_i)^\alpha}{\sum_i^k w_i} < \epsilon^\alpha \tag{5}$$

and take the α^{th}-root of both sides to get

$$D_{\mathcal{G}}(X) > \epsilon \tag{6}$$

- For α positive, the proof is identical except that neither reversal of the inequality applies.

Therefore, for every object x such that $D_{\mathcal{G}}(x) \leq \epsilon$, $\exists g_i \in \mathcal{G}$ such that $d(g_i, x) \leq \epsilon$. This means that such objects x must be accepted by at least one individual range query, and that all must therefore be within the union of the range queries. **QED**.

This theorem is important, as it shows that we can use existing indexing methods without fear of false dismissals. A post-processing step can then check every member of the union and discard any whose actual aggregate dissimilarity exceeds the threshold.

4 Experimental Setup

The core parts of FALCON were implemented in C, C++ and Perl, and tested on Intel Pentium II^{TM} workstations under Linux. As noted before, Figure 2 shows a screenshot of one implementation in action.

We tested FALCON with the intent of answering five central questions.

(a) Does FALCON "learn" to model concave and disjunctive queries?
(b) Does FALCON provide satisfactory precision/recall?
(c) Does FALCON rapidly converge to a satisfactory level of precision/recall?
(d) What is a suitable value of α?
(e) How fast is FALCON?

Four data sets were used during the experiments, each with exactly one query. Two of the data sets were synthetic, and two consisted of real data. We used the standard Euclidean distance as the pairwise distance metric in d each case; with the $2D\_20K$ data set, we also experimented with L_∞.

2D_50K: This synthetic data set consists of 50,000 points in 2-dimensional Cartesian space, randomly distributed approximately uniformly within the axis-aligned square (-2,-2) - (2,2).

2D_20K: This synthetic data set was generated using identical rules to that of the 2D_50K data set, but consists only of 20,000 points.

PEN: This database was obtained from the UCI repository [8], and consists of objects that correspond to handwritten digits, with features being the classification of each and the coordinates of spatially re-sampled points. We used the existing split of 3498 objects in the training set and 7494 in the test set.

STOCKS: Daily closing prices for up to five year periods were collected for 51 stocks from Yahoo's online quote server. They were split into 1856 non-overlapping vectors of length 32, which were then processed via the discrete wavelet transform (DWT). All 32 coefficients for each vector were used for L_2 distance computations.

4.1 Queries

Each data set was paired with a corresponding implicit user query which was used to objectively label each example as positive or negative. These were used in the simulation, such that no actual human intervention was required during testing. The only other instance in which this information was available to the system was for evaluation purposes, as discussed in the next subsection.

In each tests, we had three sets of objects. One, the seed set, consisted of five objects which determine the initial query. A second, larger set was ranked in each iteration and used to provide feedback. A third and largest set was also ranked, but not used for feedback. The limiting of feedback simulates a sampling approach; we show that in many cases, "training" or query refinement could be done on a small, hopefully representative subset of the whole in order to limit training time while preserving expressiveness and accuracy. It may even be feasible to provide the user a hierarchical view of a database – perhaps traversing something like an M-tree – in order to allow the user to easily select examples [3]. Results on the testing set thus allow us to empirically determine whether such sampling methods may be fruitful.

In addition, we also varied the values of α. We had little theoretical reason to believe that one value of α would be optimal for all queries on all data sets, and therefore tried multiple

values.

RING: We paired this query with the 2D_50K data set. Points were marked as positive examples if and only if they were between 0.5 and 1.5 units from the origin, inclusive. We limited feedback to only a subset of 1000 points. 431 points in the subset and 19,734 points in the full set met this criterion. The five seeds were generated 1.4 units from the origin at integral multiples of 72 degrees counter-clockwise from the positive X axis. This is a fairly simple query, in that its only notable feature is the unsuitable region within the middle. However, *none* of the previous methods handle this correctly, as they expect contiguous, convex regions.

TWO_CIRCLES: Points from 2D_20K were considered positive if and only if they were within 0.5 units of either (-1,-1) or (1,1); 1899 points qualified. The subset consisted of 1000 points randomly drawn from the full set, including 99 positive examples. The seeds were randomly generated from within the two circles.

This query was designed to test FALCON's ability to handle disjunctive queries. Clearly, any system designed to assume contiguous isosurfaces will have difficulty with this query.

PEN: For the query over the PEN data set, we chose to consider as positive those examples that had been classified as 4's. 364 vectors in the small set and 780 in the full set were positive instances. The five seeds were drawn randomly from the positive instances in the training set.

The intent here is to show that FALCON can answer a reasonable query on real data.

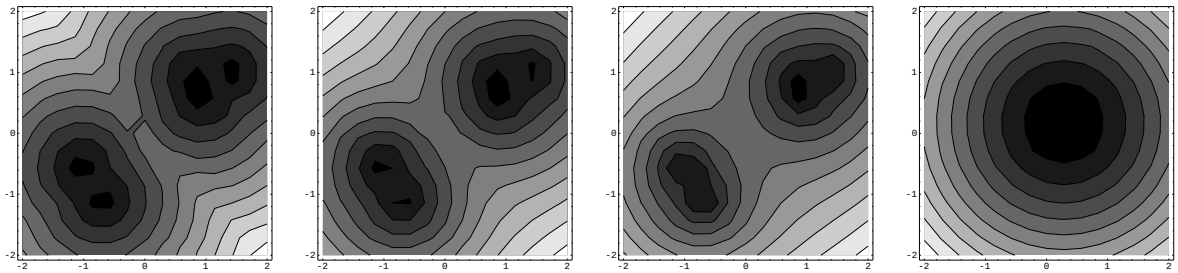
STOCKS: The query over the STOCKS data set looks for approximately "flat" stocks. Hence, we classified examples as positive if and only if the slopes of their least-squares linear approximations were within the range -0.02 to 0.02. We again limited training feedback to occur only on a subset; this consisted of 500 examples. 153 stocks in the subset and 540 in the full set were considered "flat" by this criterion. The five seeds were the vectors of DWT coefficients of five perfectly flat lines with y-intercepts of 8, 40, 80, 200 and 400, chosen in order to allow for variance in the y-intercept, while emphasizing the slope.

Like the PEN query, this is also helpful for evaluating performance on real data with a reasonable query.

4.2 Evaluation Methodology

We now outline the methodology we used for evaluating FALCON. With each data set and its paired query, we tested with multiple values of α: $-\infty$, which essentially scores by minimum distance; -100, which approximates that, but risks problems related to finite precision; -10; -5; -2; +2; and +5.

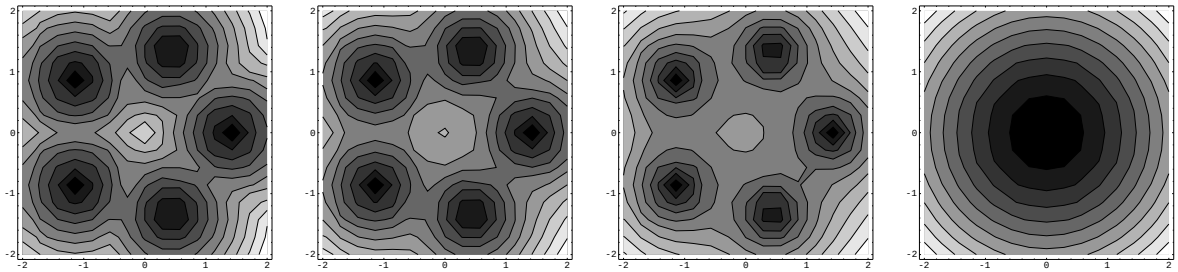
Figure 3 shows contour plots for four values of α in the TWO_CIRCLES. Note that $\alpha = +2$ reflects only the center of the "good set", even on the TWO_CIRCLES query, and hence is



$\alpha = -\infty$ $\alpha = -5$ $\alpha = -2$ $\alpha = +2$

Figure 3: Contour plots for the seeds of the TWO_CIRCLES query.

expected to fail on such a disjunctive query.



$\alpha = -\infty$ $\alpha = -5$ $\alpha = -2$ $\alpha = +2$

Figure 4: Contour plots for the seeds of the RING query.

Figure 4 shows contour plots for four values of α in the RING. Again, $\alpha = +2$ stands out with contours that are not going to rank the points very well.

FALCON ranks objects by computed score; to evaluate the rankings, we measure precision/recall. We define recall and precision as follows.

We can consider subsets of the ranked list, starting with the best, and note how many examples we have to examine in this list before we include a certain percentage of the actual positive examples with the training set. The latter percentage is defined as "recall"; the ratio of these positive examples versus the minimum size of that subset, anchored at the top is defined as "precision". Hence, the optimal score of 100% precision can only be achieved for a given level of recall if that percentage of positive examples actually appears contiguously in the highest ranks of the list. Following the typical practice in information retrieval, we compute the precision for 10 levels of recall equally spaced from 10% to 100%.

With each combination of a query, and a value of α, we used the following procedure:

- Begin with the "good set" consisting of only the seeds. Compute precision/recall over the full set for all 10 levels. This gives us a baseline that varies only with the contours generated by α.
- Repeat the following as needed:
  - Score the examples within the sample, using the current $D_{\mathcal{G}}$ function.
  - Find the top twenty for which simulated feedback has never been given. Twenty was arbitrarily chosen as an approximate upper limit for what a human operator might tolerate. Presenting more would reduce the iteration count, but might also inconvenience a user. In addition, a good sampling algorithm should reduce the number that need to be presented simultaneously.
  - Simulate feedback, and add any newly-found positive examples into the "good set" $\mathcal{G}$, which therefore modifies $D_{\mathcal{G}}$.
  - Repeat the precision/recall procedure on the full set, but with the possibly modified $D_{\mathcal{G}}$.

4.3 Speed

We also ran range-query speed tests. The query and data involved were TWO_CIRCLES and the same subset of 2D_20K used previously, with corresponding seeds. For the indexing structure, we used M-trees [3]. Sequential scanning served as a baseline with which to compare the method described in Theorem 1.

For our first assessment, we varied the threshold ϵ while measuring elapsed time and computational cost. Second, we varied the seeds with constant threshold while using the same measures.

5 Results and Discussion

We present our analysis of the results, organized with regard to the five central questions.

The following notes apply to the various graphs. For all the figures marked "Precision versus Recall", such as Figures 5, 7, and 10, one line is plotted per charted iteration. Not all iterations were charted, for purposes of readability. Each line is drawn with ten points, each of which shows precision at one level of recall from 10% to 100% at 10% increments. Hence, the general trend of the lines tracks the progress of FALCON as feedback is provided on increasing numbers of points.

Any figure that tracks "Precision versus Iterations", such as Figures 8 and 9, instead focuses on precision at one level of recall, 40%, for multiple values of α. Positive slopes indicate positive progress. 40% was arbitrarily chosen as a level at which it is non-trivial, but also not extremely difficult, to provide good precision.

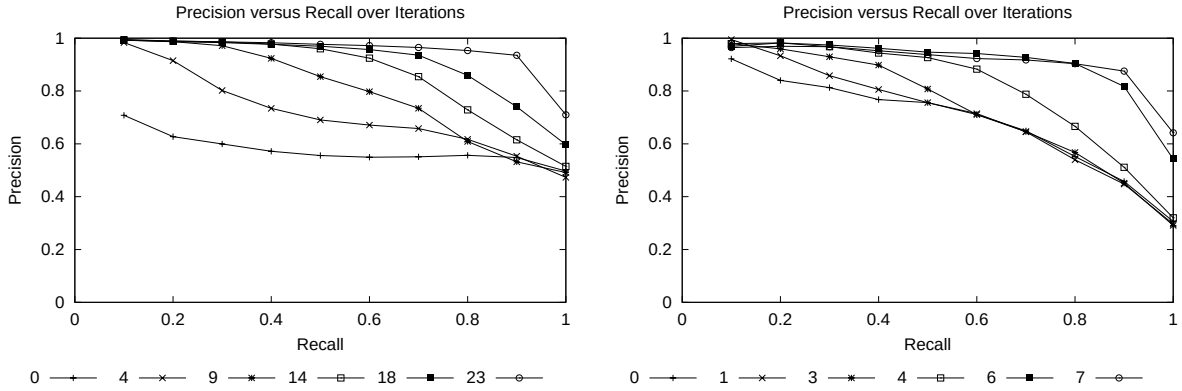
Figures 11 and 12, labelled "Precision at Multiple Levels of α" also track precision, but after 5 and 20 iterations. The first shows precision at 40% recall; the second, average precision based on all 10 levels of recall.

Recall is always expressed as a ratio from 0 to 1 indicating what proportion of the database is being considered; 0.2, for example, would mean the top 20% as rated by aggregate dissimilarity.

Precision is also expressed as such a ratio, indicating the percentage of "good" objects at the given level of recall.

5.1 Concave and Disjunctive Queries

The RING query is concave, but contiguous; the TWO_CIRCLES query is disjunctive. As one sees in Figure 5, FALCON can handle either data set with high levels of precision versus recall. This is a substantial improvement over existing methods such as MARS and MindReader, which would have difficulty with these two.



RING TWO_CIRCLES

Figure 5: Precision versus recall for $\alpha = -5$, for RING and TWO_CIRCLES, using Euclidean pairwise distance.

The numbers in the legend indicate the number of feedback iterations to achieve that level of progress. We observe that in both cases, precision largely stabilizes after the first several iterations – especially for TWO_CIRCLES.

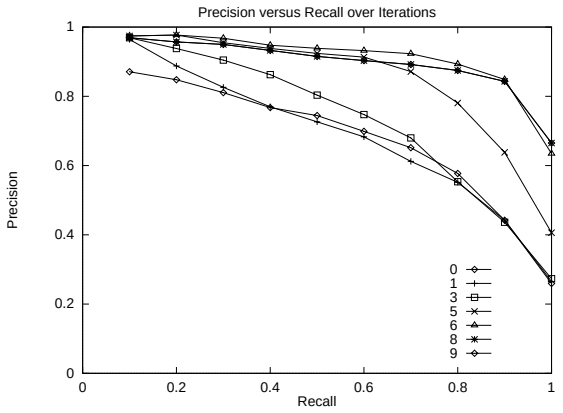


Figure 6: Precision versus recall for $\alpha = -5$, TWO_CIRCLES, and L_∞ as distance metric d

The success on the TWO_CIRCLES query is particularly notable because that set is completely disjunctive; there are two distinct regions of points that deserve high rankings, separated by points that do not. To further test the system, we ran the same query substituting L_∞ for the Euclidean distance as pairwise metric d; Figure 6 shows the resulting precision-recall.

5.2 Quality of Results on Real Data

Both the PEN and STOCKS queries are reasonable queries on real data. Consequently, FALCON's performance on these, as shown in Figure 7, is relevant to any question as to whether FALCON "works" on real data.

We note that, for the PEN query, convergence at all levels of recall below 100% is very fast; the rest of the iterations after the 4^{th} recall do not add precision except at the highest level of recall. The pattern with the STOCKS query is more interesting, with large gaps between the lines. One plausible explanation is that the positive examples were not evenly distributed in vector space, but instead clustered. The first member of a cluster to be added to $\mathcal{G}$ would immediately cause everything nearby to move upwards in the rankings. This is particularly plausible given that many of the stock vectors were consecutive slivers from the same stock over time, and therefore may have had similar properties.

As shown by Figure 7, FALCON provides good precision at almost all levels of recall. In particular, it provides perfect precision at most levels of recall for the PEN data set and query, identifying objects that correspond to 4's. Note also that FALCON provides good precision on the RING and TWO_CIRCLES queries, as shown in Figure 5, as cited above.

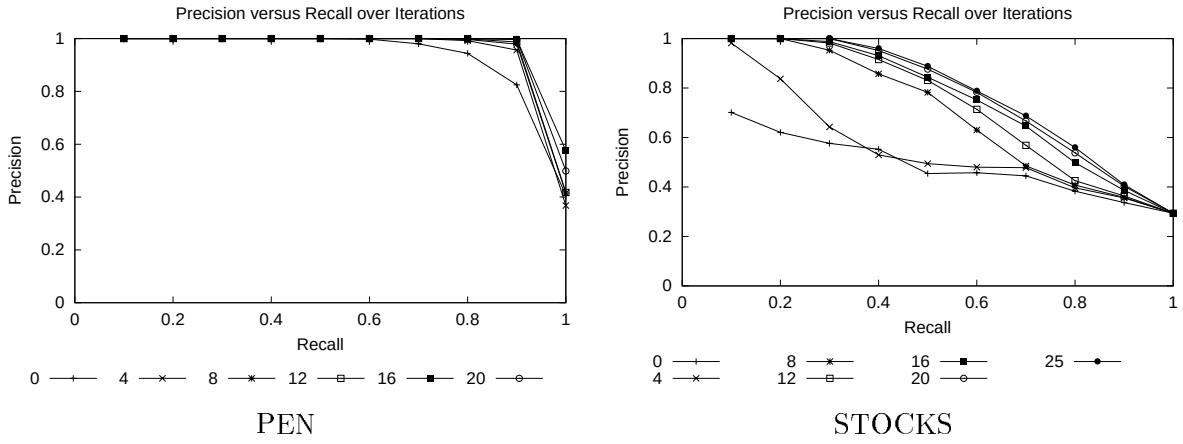


Figure 7: Precision versus recall for $\alpha = -5$, for PEN and STOCKS.

5.3 Speed of Convergence

Referring to Figures 8 and 9, which show precision at 40% recall versus the number of iterations, we see that FALCON quickly reaches good levels of precision. FALCON can attain high levels of precision over high levels of recall, even with an early $\mathcal{G}$ that has not yet grown to include many of the "good points" in even a subset of the full set.

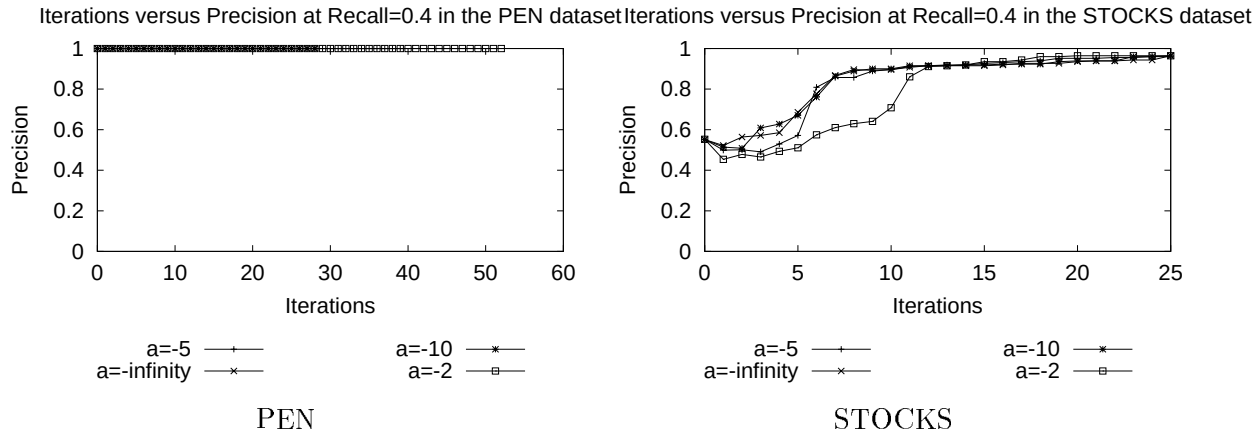
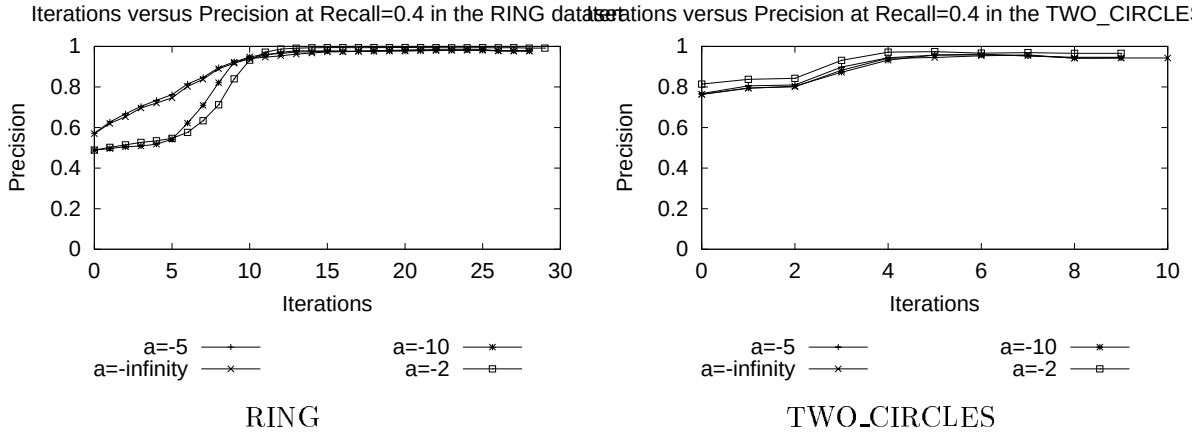


Figure 8: Precision versus iterations at recall=40%.

The STOCKS query is unlike other queries, whose trends for different values of α are far more similar to each other in their increasing until a steady state is reached. For STOCKS, there are very wide gaps in performance until 12 iterations have elapsed.



RING TWO_CIRCLES

Figure 9: Precision versus iterations at recall=40%

Depending upon the complexity of the query and the data set, progress can be very fast (such as in the PEN data set), or slower (as in the RING data set). For 40% recall, precision exceeds 90% after only 4 to 11 iterations for $\alpha = -5$ on all four queries.

There can be over-fitting, as seen in the TWO_CIRCLES data set; multiple iterations grow G without improving precision. The law of diminishing returns applies to increasing $|\mathcal{G}|$, suggesting that one case for sampling relevance feedback is that one really does not need to maximize $|\mathcal{G}|$ to attain near-optimal levels of precision.

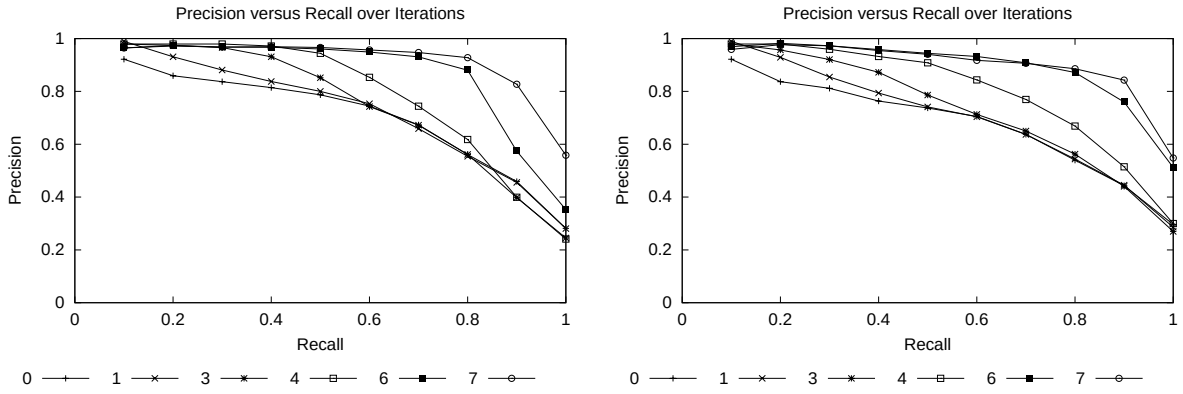
5.4 Optimal Value of α

We hypothesized that $\alpha = -5$ would be the optimal choice in our set of possible α's. We found that $\alpha = -100$ usually performed somewhat worse than $\alpha = -5$, due to approximations necessary when dealing with exponents of that magnitude on data sets with any significant range of dissimilarities.

Figure 10 show precision versus recall for the TWO_CIRCLES data set, using $\alpha = -2$ and $\alpha = -10$. As usual, the numbers in the legend indicate iterations. In both cases, all good points in the sample were added to $\mathcal{G}$ in no more than 9 iterations of feedback. We also note that the progress in precision over multiple levels of recall was fairly steady throughout.

Precision was generally similar for all negative values of α, especially once $\mathcal{G}$ had reached its maximum size. With positive values of α, precision was quite low at most levels of recall; see Figures 11 and 12 for some results.

The first figure shows precision at 40% recall for each level of tested α; each line tracks the precisions yielded by one particular value of α for the different queries. This allows us



($\alpha = -2$) ($\alpha = -10$)

Figure 10: Precision versus recall on the TWO_CIRCLES data set.

to compare α both early and late in the process. In the case of TWO_CIRCLES, whose $\mathcal{G}$ stabilized in fewer than 20 iterations, we used the final results. Observe that the differences among the negative values of α largely disappear by the 20^{th} iteration of feedback.

As we can see in the latter figure, difference in average precision over all levels of recall after 20 iterations are a bit more marked than at 40%, but again the negative values of α yield fairly similar average precision.

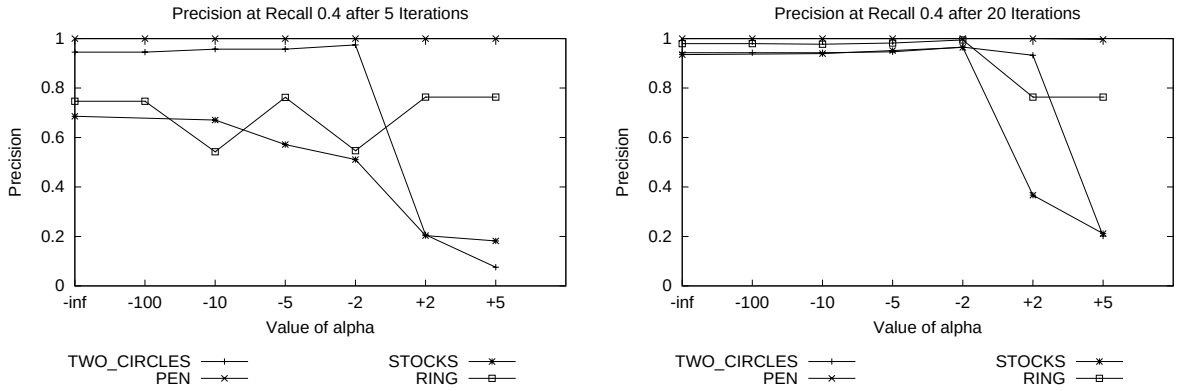
We still favor $\alpha = -5$, as it provided good performance as empirically observed, and in no case is it significantly inferior. Similar values appear to yield similar results.

5.5 Speed

Empirically, we have found that merging the results of k separate range queries as per Theorem 1 is satisfactory in terms of both the number of distance computations, and the elapsed time. Some of these results may be noted in Figures 13 and 14.

The two graphs in Figure 13 compare the elapsed (real) time and the pairwise distance computation costs incurred in searching the TWO_CIRCLES training data set for points within a variable aggregate dissimilarity threshold from the seeds. These results demonstrate that the k range queries can be merged into one.

We note that the cross-over point where sequential scan performs as quickly as merging occurs at a threshold of approximately 0.7; checking Figure 16 shows that such a threshold accepts approximately 20% of the database. It is our belief that users of large interactive databases will rarely be interested in queries of such breadth, and thus the merging method is worthwhile.



After 5 Iterations

After 20 Iterations

Figure 11: Precision at 40% recall

As shown in Figure 14, both measurements of performance cost appear to scale with the number of seeds. These results are taken without caching the distance computations from previous searches; these searches were all independent. These two graphs compare the elapsed (real) time and the pairwise distance computation costs incurred in searching the TWO_CIRCLES training data set for points with aggregate dissimilarity less than or equal to 1, varying the number of seeds.

Figure 15 shows results for the same test, but with the underlying distance function $d = L_\infty$, not $d = L_2$. The trends are quite similar.

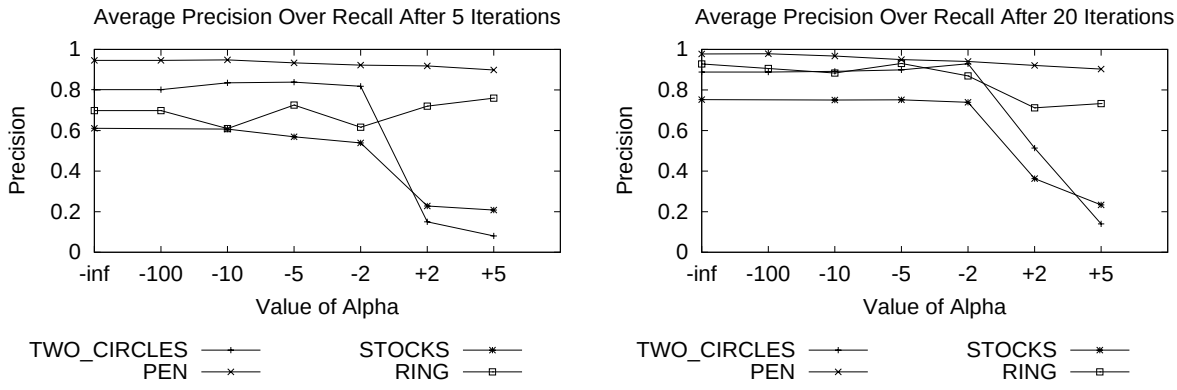
These results, of course, are tied to the characteristics of the data set and the query. Figure 16 describes the inclusiveness of the query based on threshold for both pairwise distance functions The Y axis indicates how many objects, out of a total of 1000, had a $D_{\mathcal{G}}$ of less than the threshold.

No direct experimental comparisons are shown with previous methods. This is because our queries were specifically designed to include queries of disjunctive and other highly non-convex behavior in general metric spaces, and thus previous methods simply do not apply.

6 Conclusions

We proposed a method to handle queries by multiple examples, on arbitrary vector or metric databases.

Our method applies to general metric spaces, as the distance combination function depends on only the pairwise distances and not the actual nature of the data. In addition, it handles



After 5 Iterations

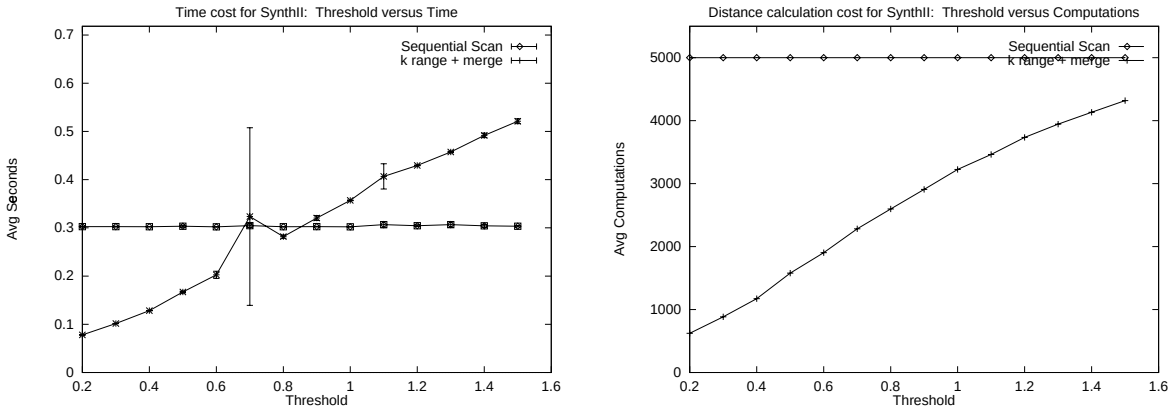
After 20 Iterations

Figure 12: Average precision at multiple values of α.

disjunctive and concave queries that are fundamentally *impossible* for traditional relevance feedback methods. Using our method, a user could, without any domain-specific query language, specify a disjunctive query merely by labelling as "good" objects representative of the different classes. We argue that this combination of general applicability, power and ease-of-use makes this method more valuable than other existing systems today.

The heart of our method is the FALCON aggregate dissimilarity, or $D_{\mathcal{G}}$, which is able to "learn" disjunctive queries via relevance feedback. Additional contributions include the following:

- Theorem 1, which shows that we can use indexing structures that support range queries, to speed up our search, guaranteeing zero false dismissals.
- Experiments on real and synthetic data, that show that the proposed method ("FALCON") achieves good precision and recall. For instance, with all queries, $\alpha = -5$ yielded at least 80% precision at 50% recall with 10 iterations.
- Experiments that show that FALCON needs at most 10 feedback iterations to reach high precision/recall, and will reach a "steady state" for all levels of recall below 100% within approximately 20 iterations. Beyond 20 iterations, minor progress is made for 100% recall.
- Experiments that show that a good range for α includes -10 to -2; on many queries, the sensitivity of the performance on α is low.



Elapsed time

Distance Computations

Figure 13: Time and computational cost for TWO_CIRCLES, varying threshold.

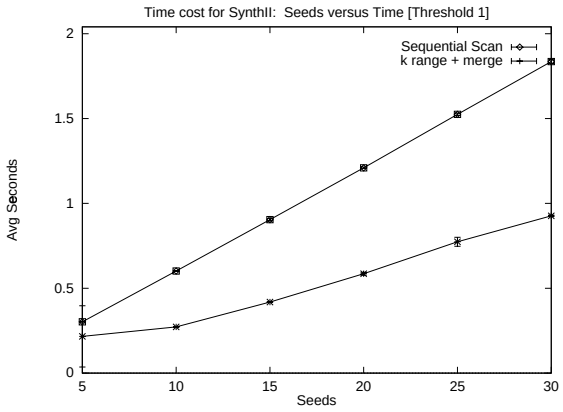
- Experiments that show that we can use the method described by Theorem 1 to gain up to 200% observed performance improvements versus sequential scanning.

Possible avenues for future work include using $D_{\mathcal{G}}$ for other data mining tasks, such as classification and representative selection [7].

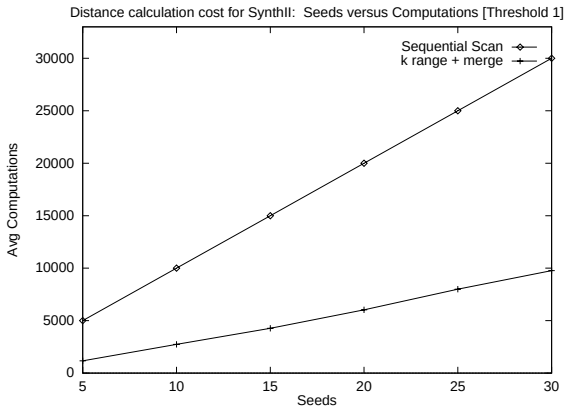
Acknowledgements

This research was partially funded by the National Science Foundation under Grants No. IRI-9625428, DMS-9873442, and IIS-9910606; also, by the National Science Foundation, ARPA and NASA under NSF Cooperative Agreement No. IRI-9411299; DARPA/ITO through Order F463, issued by ESC/ENS under contract N66001-97-C-851; and the Office of Naval Research Grant N-00014-96-1-1222. Additional funding was provided by donations from NEC and Intel. Views and conclusions contained in this document are those of the authors and should not be interpreted as representing official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or of the United States Government.

All trademarks are property of their respective owners.

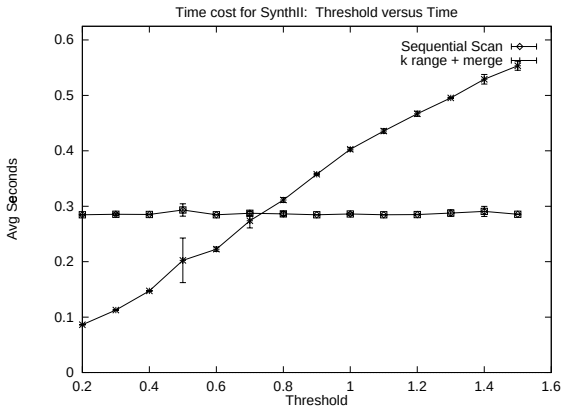


Elapsed time

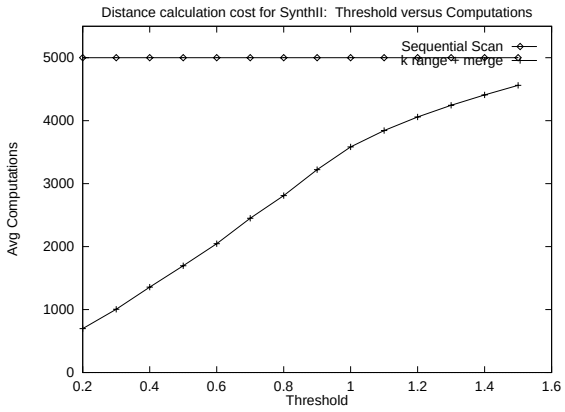


Distance Computations

Figure 14: Time and computational cost for TWO_CIRCLES, varying seeds.

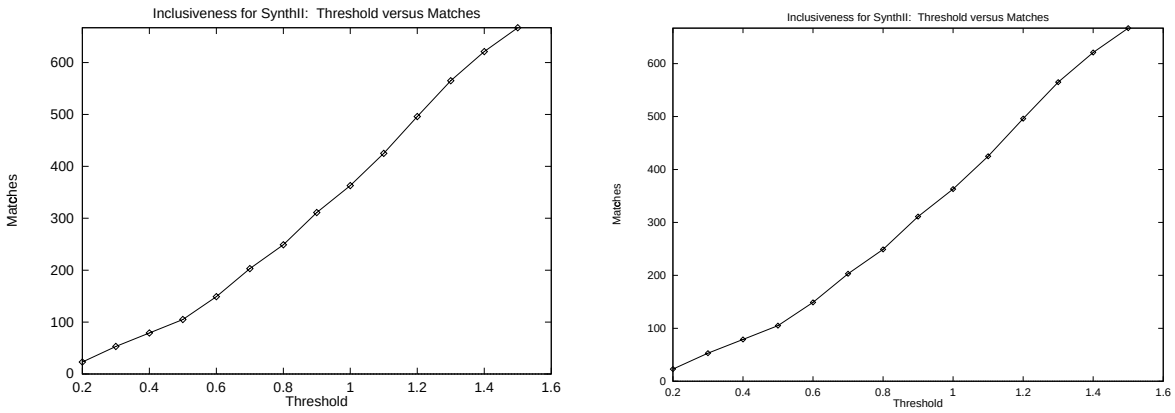


Elapsed time



Distance Computations

Figure 15: Time and computational cost on TWO_CIRCLES, with L_∞.



L_2 L_∞

Figure 16: Threshold versus Selectivity, with $d = L_2$ and $d = L_\infty$.

A Appendix

A.1 Intuition Behind the Combination Function

Let $d_i = d(x, g_i)$, for $i = 1..n$, be the n pairwise distances of object x from each object g_i in the selected set $\mathcal{G}$. It is easier to justify Equation 1 if we use similarities instead of distances. Let

$$s_i = s(x, g_i),\ i = 1..n \tag{7}$$

be the pairwise similarity values for the object x and each object in $\mathcal{G}$. Also, let $S_{\mathcal{G}}(x)$ denote the similarity (or "desirability") of object x with respect to the selected objects in $\mathcal{G}$. Let these aggregate similarities take positive values, with higher values meaning higher desirability.

We can link similarities with distances with any monotonically decreasing function; see [6] for some options. Using the simplest option

$$s_i = \frac{1}{d_i} \tag{8}$$

and

$$S_{\mathcal{G}}(x) = \frac{1}{D_{\mathcal{G}}(x)} \tag{9}$$

we see that Equation 1 is, implicitly, combining similarities in a fashion related to a fuzzy "OR" as discussed in [14].

References

[1] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R*-tree: An efficient and robust access method for points and rectangles. *ACM SIGMOD*, pages 322–331, May 23-25 1990.

[2] Thomas Brinkhoff, Hans-Peter Kriegel, and Bernhard Seeger. Efficient processing of spatial joins using R-trees. In *Proc. of ACM SIGMOD*, pages 237–246, Washington, D.C., May 26-28 1993.

[3] Paolo Ciaccia, Marco Patella, and Pavel Zezula. M-tree: An efficient access method for similarity search in metric spaces. *VLDB*, pages 426–435, 1997.

[4] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis.* John Wiley and Sons, 1973.

[5] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *Proc. ACM SIGMOD*, pages 47–57, Boston, Mass, June 1984.

[6] Yoshiharu Ishikawa, Ravishankar Subramanya, and Christos Faloutsos. Mindreader: Querying databases through multiple examples. Technical report, 1998.

[7] M.V. Boland M.K. Markey and R.F. Murphy. Towards objective selection of representative microscope images. *Biophys. J.*, 1999. in press.

[8] P.M. Murphy and D.W. Aha. UCI Repository of Machine Learning Databases. Department of Information and Computer Science, University of California, Irvine, CA. [http://www.ics.uci.edu/~mlearn/MLRepository.html], 1994.

[9] T.R. Payne. *Dimensionality Reduction and Representation for Nearest Neighbour Learning.* PhD thesis, The University of Aberdeen, Scotland, 1999.

[10] Kriengkrai Prokaew, Sharad Mehrotra, Michael Ortega, and Kaushik Chakrabarti. Similarity search using multiple examples in mars. In *1999 International Conference on Visual Information Systems*, June 1999.

[11] Joseph John Rocchio. Relevance feedback in information retrieval. In Gerard Salton, editor, *The SMART Retrieval System – Experiments in Automatic Document Processing*, pages 313–323. Prentice Hall, Englewood Cliffs, N.J., 1971.

[12] Yong Rui, Thomas S. Huang, and Sharad Mehrotra. Content-based image retrieval with relevance feedback in MARS. In *Proceedings of IEEE International Conference on Image Processing '97*, Santa Barbara, CA, October 1997.

[13] Yong Rui, Thomas S. Huang, and Sharad Mehrotra. Human perception subjectivity and relevance feedback in multimedia information retrieval. In *Proceedings of IS&T and SPIE Storage and Retrieval of Image and Video Databases VI*, San Jose, CA, January 1998.

[14] G. Salton, E.A. Fox, and H. Wu. Extended boolean information retrieval. *CACM*, 26(11):1022–1036, November 1983.

[15] C.T. Zahn. Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Trans. on Computers*, C-20(1):68–86, January 1971.