Distributed Ranking Methods for Geographic Information Retrieval*

Marc van Kreveld, Iris Reinbacher, Avi Arampatzis, and Roelof van Zwol

Institute of Information and Computing Sciences, Utrecht University
P.O.Box 80.089, 3508 TB Utrecht, The Netherlands
marc@cs.uu.nl, iris@cs.uu.nl, avgerino@cs.uu.nl, roelof@cs.uu.nl

Summary. Geographic Information Retrieval is concerned with retrieving documents that are related to some location. This paper addresses the ranking of documents by both textual and spatial relevance. To this end, we introduce *distributed ranking*, where similar documents are ranked spread in the list instead of consecutively. The effect of this is that documents close together in the ranked list have less redundant information. We present various ranking methods, efficient algorithms to implement them, and experiments to show the outcome of the methods.

1 Introduction

The most common way to return a set of documents obtained from a Web query is by a ranked list. The search engine attempts to determine which document seems to be the most relevant to the user and will put it first in the list. In short, every document receives a *score*, or *distance to the query*, and the returned documents are sorted by this score or distance.

There are situations where the sorting by score may not be the most useful one. When a more complex query is done, composed of more than one query term or aspect, documents can also be returned with two or more scores instead of one. This is particularly useful in *geographic information retrieval* (Jones et al. 2002, Rauch et al. 2003, Visser et al. 2002). For example, the Web search could be for castles in the neighborhood of Koblenz, and the documents returned ideally have a score for the query term "castle" and a score for the proximity to Koblenz. This implies that a Web document resulting from this query can be mapped to a point in the 2-dimensional plane.

A cluster of points in this plane could be several documents about the same castle. If this castle is in the immediate vicinity of Koblenz, all of these documents would be ranked high, provided that they also have a high score on the term "castle". However, the user probably also wants documents about other castles that may be a bit further away, especially when these documents

* This research is supported by the EU-IST Project No. IST-2001-35047 (SPIRIT).

are more relevant for the term "castle". To incorporate this idea in the ranking, we introduce *distributed ranking* in this paper. We present various models that generate ranked lists that have diversity. We also present efficient algorithms that compute the distributed rankings. To keep server load low, it is important to have efficient algorithms.

There are several reasons to rank documents according to more than one score. For example we could distinguish between the scores of two textual terms, or a textual term and metadata information, or a textual term and a spatial term, and so on. A common example of metadata for a document is the number of hyperlinks that link to that document; a document is probably more relevant if there are many links to it. In all of these cases we get two scores which need to be combined for a ranking.

In traditional information retrieval, the two scores of each document would be combined into a single score (e.g., by a weighted sum or product) which produces the ranked list by sorting. Besides the problem that it is unclear how the two scores should be combined, it also makes a distributed ranking impossible. Two documents with the same combined score could be similar documents or quite different. If two documents have two scores that are the same, one has more reason to suspect that the documents themselves are similar than when two documents have one (combined) score that is the same.

The topic of geographic information retrieval is studied in the SPIRIT project (Jones et al. 2002). The idea is to build a search engine that has spatial intelligence because it will understand spatial relationships like *close to*, *to the North of*, *adjacent to*, and *inside*, for example. The core search engine will process a user query in such a way that both the textual relevance and the spatial relevance of a document is obtained in a score. This is possible because the search engine will not only have a term index, but also a spatial index. These two indices provide the two scores that are needed to obtain a distributed ranking. The ranking study presented here will form part of the geographic search engine to be developed for the SPIRIT project. Related research has been conducted in (Rauch et al. 2003), which focuses on disambiguating geographic terms of a user query. The disambiguation of the geographic location is done by combining textual information, spatial patterns of other geographic references, relative geographic references from the document itself, and population heuristics from a gazetteer. This gives the final value for geoconfidence. The georelevance is composed of the geoconfidence and the emphasis of the place name in the document. The textual relevance of a document is computed as usual in information retrieval. Once both textual and geographic relevance are computed, they are combined by a weighted sum.

Finding relevant information and at the same time trying to avoid redundancy has so far mainly been addressed in producing summaries of one or more documents. (Carbonell and Goldstein 1998) uses the maximal marginal relevance (MMR), which is a linear combination of the relevance of the document to the user query and its independence of already selected documents.

MMR is used for the reordering of documents. A user study has been performed in which the users preferred MMR to the usual ranking of documents. This paper contains no algorithm how to actually (efficiently) compute the MMR. Following up on this, a Novelty Track of TREC (Harman 2002) discusses experimenting with ranking of textual documents such that every next document has as much additional information as possible.

(Goldstein et al. 1999) proposes another scoring function for summarizing text documents. Every sentence is assigned a score combined of the occurence of statistical features and on the occurrence of linguistic features. They are combined linearly with a weighting function. In (Goldstein et al. 2000), MMR is refined and used to summarize multiple documents. Different passages or sentences respectively are assigned a score instead of full documents.

The remainder of this paper is organized as follows. In Section 2 we present several different ranking methods and the algorithms to compute them. In Section 3 we show how the ranking methods behave on real-world data. In the conclusions we mention other research and experiments that we have done or we are planning to do.

2 Distributed Ranking Methods

In this section we will present specific ranking methods. Like in traditional information retrieval, we want the most relevant documents to appear in the ranking, while avoiding that documents with similar information appear close to documents already ranked.

We will focus on the two dimensional case only, although in principle the idea and formulas apply in higher dimensions too. We assume that a Web query has been conducted and a number of relevant documents were found. Each document is associated with two scores, for example a textual score and a spatial score (which is the case in the SPIRIT search engine). The relevant documents are mapped to points in the plane, and the query is also mapped to a point. We perform the mapping in such a way that the query is a point Q at the origin, and the documents are mapped to points $p_1, \ldots, p_n$ in the upper right quadrant, where documents with high scores are points close to Q. We can now formulate the two main objectives for our ranking procedure:

1. **Proximity to query:** Points close to the query Q are favored.
2. **Spreading:** Points farther away from already ranked points are favored.

A ranking that simply sorts all points in the representation plane by distance to Q is optimal with respect to the first objective. However, it can perform badly with respect to the second. Selecting a highly distributed subset of points is good with respect to the second objective, but the ranked list would contain too many documents with little relevance early in the list. We therefore seek a compromise where both criteria are considered simultaneously. Note

that the use of a weighted sum to combine the two scores into one makes it impossible to obtain a spreaded ranking.

The point with the smallest Euclidean distance to the query is considered the most relevant document and is always first in any ranking. The remaining points are ranked with respect to already ranked points. At any moment during the ranking, we have a subset $R \subset P$ of points that have already been ranked, and a subset $U \subset P$ of points that are not ranked yet. We choose from U the "best" point to rank next, where "best" is determined by a *scoring function* that depends on both the distance to the query Q and the set R of ranked points. Intuitively, an unranked point has a higher added value or relevance if it is not close to any ranked points. For every unranked point p,

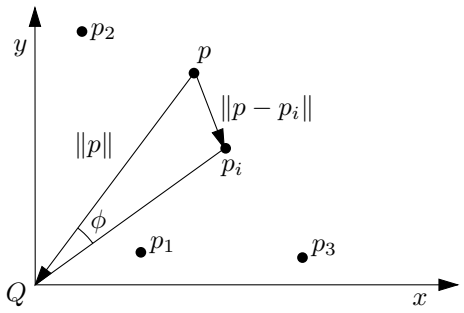

Fig. 1. An unranked point p amidst ranked points p_1, p_2, p_3, p_i, where p is closest to p_i by distance and by angle.

we consider only the closest point $p_i \in R$, where closeness is measured either in the Euclidean sense, or by angle with respect to the query point Q. This is illustrated by $\|p - p_i\|$ and ϕ, respectively, in Figure 1. Using the angle to evaluate the similarity of p and p_i seems less precise than using the Euclidean distance, but it allows more efficient algorithms, and certain extensions of angle-based ranking methods give well-distributed results.

2.1 Distance to query and angle to ranked

Our first ranking method uses the angle measure to obtain the similarity between an unranked point and a ranked point. In the triangle $\triangle pQp_i$ (see Figure 1) consider the angle $\phi = \phi(p, p_i)$ and rank according to the score $S(p, R) \in [0, 1]$, which can be derived from the following normalized equation:

$$S(p, R) = \min_{p_i \in R} \left(\frac{2(\phi(p, p_i) + c)}{\pi + 2c} \cdot \left(\frac{1}{1 + \|p\|} \right)^k \right) \tag{1}$$

Here, k denotes a constant; if $k < 1$, the emphasis lies on the distribution, if $k > 1$, we assign a bigger weight to the proximity to the query. The additive constant $0 < c \ll 1$ ensures that all unranked points $p \in N$ are assigned an angle dependent factor greater than 0. The score $S(p, R)$ necessarily lies between 0 and 1, and is appropriate if we do not have a natural upper bound on the maximum distance of unranked points to the query. If such an upper bound was available, there are other formulas that give normalized scores.

During the ranking algorithm, we always choose the unranked point p that has the highest $S(p, R)$ score and rank it next. This implies an addition to the set R, and hence, recomputation of the scores of unranked points may be necessary. We first give a generic algorithm with a running time of $O(n^2)$.

Algorithm 1: Input: A set P with n points in the plane.

1. Rank the point r closest to the query Q first. Add it to R and delete it from P.
2. For every unranked point $p \in P$ do
   a) Store with p the point $r \in R$ with the samallest angle to p
   b) Compute the score $S(p, R) = S(p, r)$
3. Determine and choose the point with the highest score $S(p, R)$ to be next in the ranking; add it to R and delete it from P.
4. Compute for every point $p' \in P$ the angle to the last ranked point p. If it is smaller than the angle to the point stored with p', then store p with p' and update the score $S(p', R)$.
5. Continue with step 3 as long as there are unranked points.

The first four steps all take linear time. As we need to repeat steps 3 and 4 until all points are ranked, the overall running time of this algorithm is $O(n^2)$. It is a simple algorithm, and can be modified to work for different score and distance functions. In fact, it can be applied to all the ranking models that will follow.

Theorem 1. *A set of n points in the plane can be ranked according to the distance-to-query and angle-to-ranked model in $O(n^2)$ time.*

If we are only interested in the top 10 documents of the ranking, we only need $O(n)$ for the computation. More generally, the top t documents are determined in $O(tn)$ time.

2.2 Distance to query and distance to ranked

In the last section we ranked by angle to the closest ranked point. It may be more natural to consider the Euclidean distance to the closest ranked point instead. In the triangle $\triangle pQp_i$ of Figure 1, take the distance $\|p - p_i\|$ from p to the closest ranked point p_i and rank according to the outcome of the following equation:

$$S(p, R) = \min_{p_i \in R} \left(\frac{\|p - p_i\|}{\|p\|^2} \right) \tag{2}$$

The denominator needs a squaring of $\|p\|$ (or another power > 1) to assure that documents far from Q do not end up too early in the ranking, which would conflict with the proximity to query requirement. A normalized equation such that $S(p, R) \in [0, 1]$ is the following:

$$S(p, R) = \min_{p_i \in R} \left((1 - \mathrm{e}^{-\lambda \cdot \|p - p_i\|}) \cdot \frac{1}{1 + \|p\|} \right) \tag{3}$$

Here, λ is a constant that defines the slope of the exponential function. Algorithm 1 can be modified to work here as well with a running time of $O(n^2)$.

Theorem 2. *A set of n points in the plane can be ranked according to the distance-to-query and distance-to-ranked model in $O(n^2)$ time.*

2.3 Addition models

So far, our distributed methods were all based on a formula that divided angle or distance to the closest ranked point by the distance to the query. In this way, points closer to the query get a higher relevance. We can obtain a similar effect but a different ranking by adding up these values. It is not clear beforehand which model will be more satisfactory for users, so we analyze these models as well.

$$S(p, R) = \min_{p_i \in R} \left(\alpha \cdot (1 - \mathrm{e}^{-\lambda \cdot (\|p\| / \|p_{max}\|)}) + (1 - \alpha) \cdot \phi(p, p_i) \cdot \frac{2}{\pi} \right) \tag{4}$$

In this equation, p_{max} is the point with maximum distance to the query, $\alpha \in [0, 1]$ denotes a variable which is used to put an emphasis on either distance or angle, and λ is a constant that defines the base e^{λ} of the exponential function. Algorithm 1 can be modified for the addition models, but as the angle $\phi(p, p_i)$ is an additive and not a multiplicative part of the score equation, we can give algorithms with better running time.

The point set is initially stored in the leaves of a binary tree T, sorted by counterclockwise (ccw) angle to the y-axis. In every leaf of the tree we also store: (i) ccw and clockwise (cw) angle to y and x-axis respectively; (ii) the distance to the query; (iii) ccw and cw score. We augment T as follows (see e.g. (Cormen et al. 1990) for augmenting data structures): In every internal node we store the best cw and ccw score per subtree. Later in the algorithm, we additionaly store the angle of the closest ccw and cw ranked point and whether the closest ranked point is in cw or ccw direction in the root of each subtree. Furthermore, we store the best score per tree in a heap for quicker localization. As shown left in Figure 2, between two already ranked points p_1 and p_2, indicated by ℓ_1 and ℓ_2, there are two binary trees, T_1 cw and T_2 ccw of the bisecting barrier line ℓ_{12}. All the points in T_1 are closer in angle to p_1 and all the points in T_2 are closer in angle to p_2. If we insert a new point p_3 to the ranking, this means we insert a new imaginary line ℓ_3 through p_3 and we need to perform the following operations on the trees:

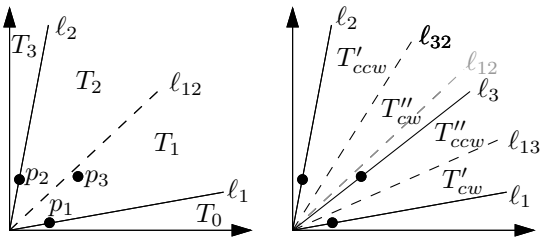

Fig. 2. The split and concatenate of trees in Algorithm 2.

1. Split T_1 and T_2 at the angle-bisectors ℓ_{32} and ℓ_{13}, creating the new trees T'_{cw} and T'_{ccw} and two intermediate trees $\overline{T}_{cw}$ and $\overline{T}_{ccw}$
2. Concatenate the intermediate trees from (1), creating one tree $\overline{T}$.
3. Split $\overline{T}$ at the newly ranked point p_3, creating T''_{cw} and T''_{ccw}.

Figure 2 right, shows the outcome of these operations. Whenever we split or concatenate the binary trees we need to make sure that the augmentation remains correct. In our case, this is no problem, as we only store the best initial scores in the inner leaves. However, we need to update the information in the root of each tree about the closest cw and ccw ranked point and the best scores. As the scores are additive, and all scores for points in the same tree are calculated with respect to the same ranked point, we simply subtract $(1-\alpha)\cdot\phi'\cdot 2/\pi$, where ϕ' denotes the cw(ccw) angle of the closest ranked point, from the cw (ccw) best score to get the new best score for the tree. We also need to update the score information in the heap. Now we can formulate an algorithm for the addition-model that runs in $O(n\log n)$ time.

Algorithm 2: Input: A set P with n points in the plane.

1. Create T with all points of P, the augmentation and a heap that contains only the point p closest to the query Q.
2. Choose the point p with the highest score $S(p,R)$ as next in the ranking by deleting the best one from the heap.
3. For every last ranked point p do:
   a) Split and concatenate the binary trees as described above and update the information in their roots.
   b) Update the best-score information in the heap:
      i. Delete the best score of the old tree T_1 or T_2 that did not contain p.
      ii. Find the four best scores of the new trees $T'_{cw}, T'_{ccw}, T''_{cw}$, and T''_{ccw} and insert them in the heap.
4. Continue with step 2.

Theorem 3. *A set of n points in the plane can be ranked according to the angle-distance addition model in $O(n\log n)$ time.*

Another, similar, addition model adds up the distance to the query and the distance to the closest ranked point:

$$S(p,R) = \min_{p_i \in R} \left(\alpha \cdot (1 - \mathrm{e}^{-\lambda_1 \cdot (\|p\| / \|p_{max}\|)}) + (1-\alpha)(1 - \mathrm{e}^{-\lambda_2 \cdot \|p - p_i\|}) \right) \quad (5)$$

Again, p_{max} is the point with maximum distance to the query, $\alpha \in [0,1]$ is a variable used to influence the weight given to the distance to the query (proximity to query) or to the distance to the closest point in the ranking (high spreading), and λ_1 and λ_2 are constants that define the base of the exponential function. Note that Algorithm 2 is not applicable for this addition model. This is easy to see, since the distance to the closest ranked point does not change by the same amount for a group of points. This implies that the score for every unranked point needs to be adjusted individually when adding a point to R. We can modify Algorithm 1 for this addition model. Alternatively, we can use the following algorithm that has $O(n^2)$ running time in the worst case, but a typical running time of $O(n \log n)$.

Algorithm 3: Input: A set P with n points in the plane.

1. Rank the point p closest to the query Q first. Add it to R and delete it from P. Initialize a list with all unranked points.
2. For every newly ranked point $p \in R$ do:
   a) Insert it to the Voronoi diagram of R.
   b) Create for the newly created Voronoi cell a list of unranked points that lie in it by taking those points that have p as closest ranked from the lists of the neighboring cells. For all $R' \subseteq R$ Voronoi cells that changed, update their lists of unranked points.
   c) Compute the point with the best score for the newly created Voronoi cell and insert it in a heap H. For all $R' \subseteq R$ Voronoi cells that changed, recompute the best score and update the heap H accordingly.
3. Choose the point with the best overall score from the heap H as next in the ranking; add it to R and delete it from P and H.
4. Continue with step 2.

Since the average degree of a Voronoi cell is six, one can expect that a typical addition of a point p to the ranked points involves a set R' with six ranked points. If we also assume that, typically, a point in R' loses a constant fraction of the unranked points in its list, we can prove an $O(n \log n)$ time bound for the whole ranking algorithm. The analysis is the same as in (Heckbert and Garland 1995, van Kreveld et al. 1997). The algorithm can be applied to all ranking methods described so far.

Theorem 4. *A set of n points in the plane can be ranked by the distance-distance addition model in $O(n^2)$ worst case and $O(n \log n)$ typical time.*

3 Experiments

We implemented the generic ranking Algorithm 1 for the basic ranking methods described in Subsections 2.1, 2.2, and 2.3. Furthermore we implemented

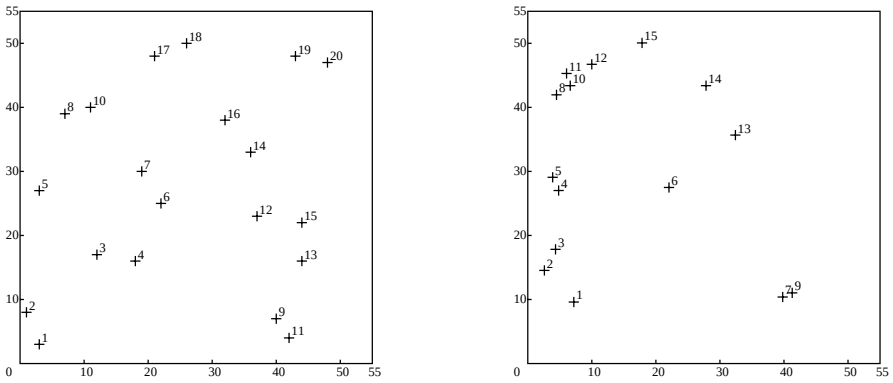

Fig. 3. Ranking by distance to origin only.

an extension called *staircase enforcement*, explained in Subsection 3.2. We compare the outcomes of these algorithms for two different point sets shown in Figure 3. The point set at the left consists of 20 uniformly distributed points, the point set at the right shows the 15 most relevant points for the query 'safari africa' which was performed on a dataset consisting of 6,500 Lonely Planet web pages. The small size of the point sets was chosen out of readability considerations.

3.1 Basic ranking algorithms

Figure 3 shows the output of a ranking by distance-to-query only. It will function as a reference point for the other rankings. Points close together in space are also close in the ranking. In the other ranking methods, see Figure 4, this is not the case anymore. This is visible in the 'breaking up' of the cluster of four points in the upper left corner of the Lonely Planet point set rankings. Note also that the points ranked last by the simple distance ranking are always ranked earlier by the other methods. This is because we enforced higher spreading over proximity to the query by the choice of parameters. The rankings are severely influenced by this choice. In our choice of parameters we did not attempt to obtain a "best" ranking. We used the same parameters in all three ranking methods to simplify qualitative comparison.

3.2 Staircase enforced ranking algorithms

In the staircase enforced methods, shown in Figure 5, the candidates to be ranked next are only those points that lie on the (lower left) staircase of the point set. The scoring functions are as before. A point p is on the staircase of a point set P if and only if for all $p' \in P \setminus \{p\}$, we have $p_x < p'_x$ or $p_y < p'_y$. So, with this limitation, proximity to the query gets a somewhat higher

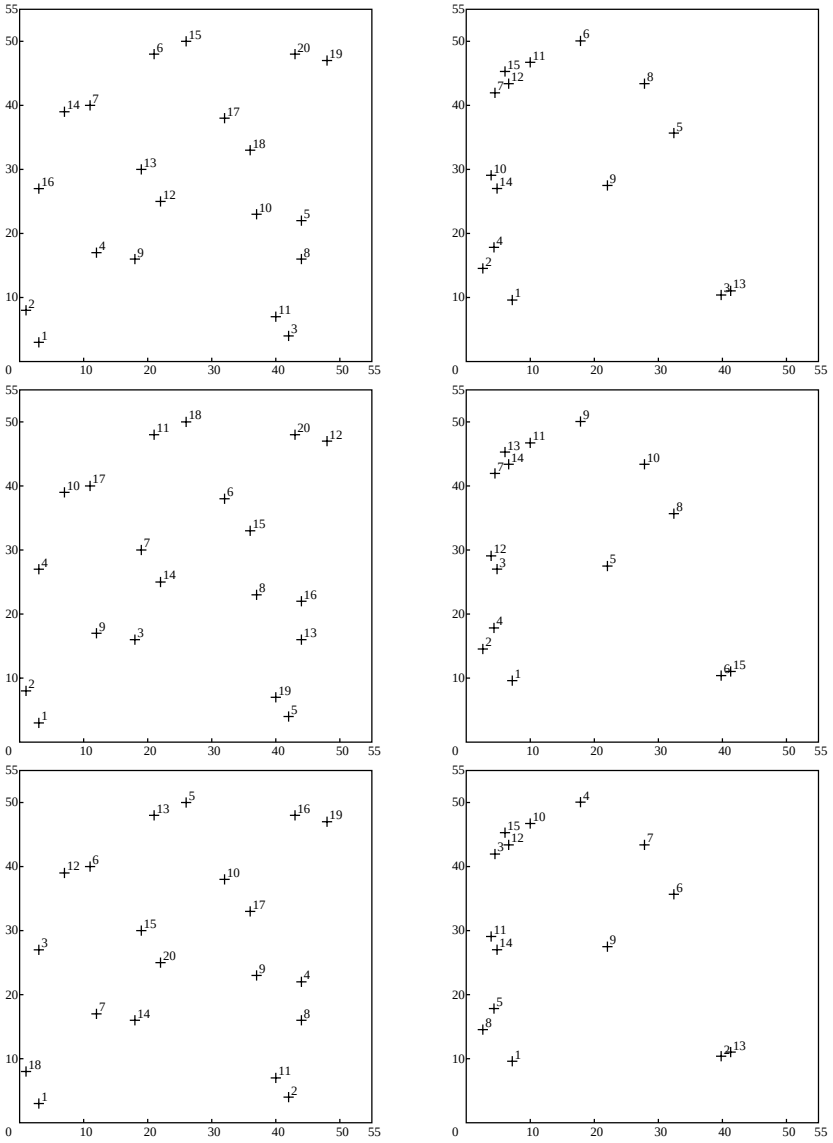

Fig. 4. Top: Ranking by distance to origin and angle to closest ($k = 1$, $c = 0.1$). Middle: Ranking by distance to origin and distance to closest (Equation 3, $\lambda = 0.05$). Bottom: Ranking by additive distance to origin and angle to closest ($\alpha = 0.4$, $\lambda = 0.05$).

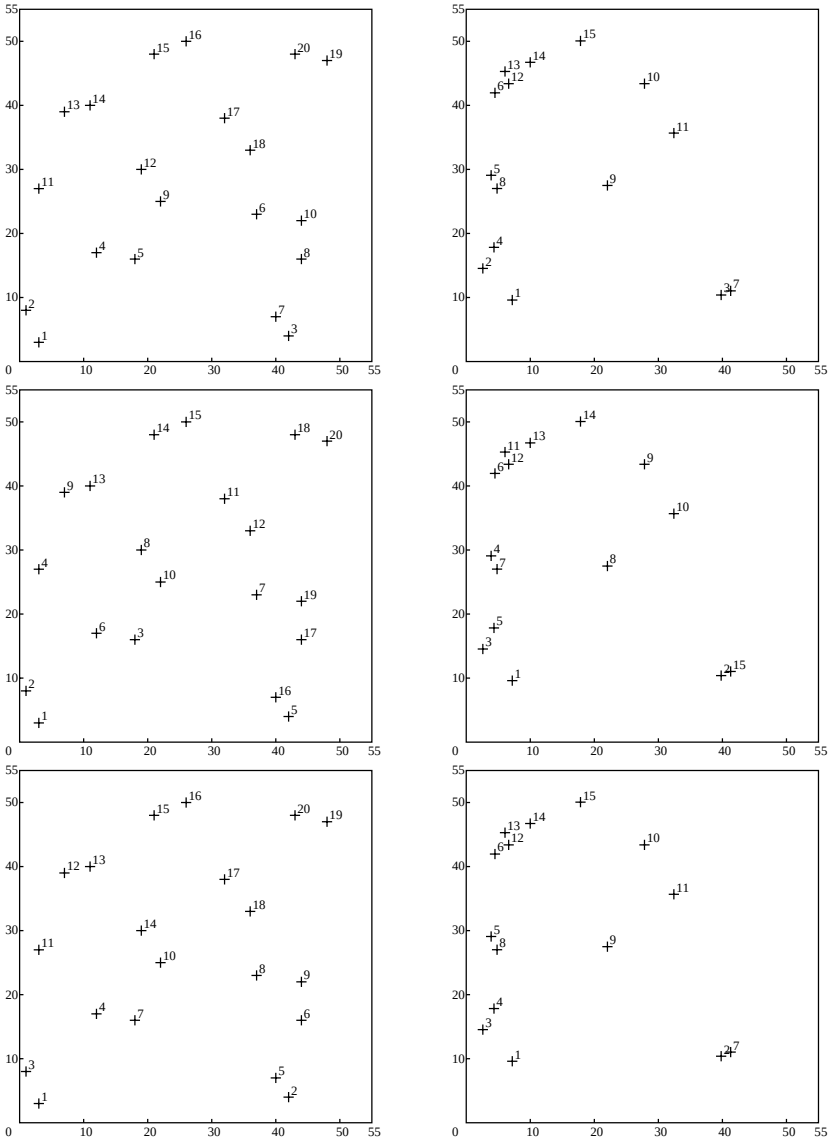

Fig. 5. Same as Figure 4, but now staircase enforced.

importance compared to the basic algorithms, which is clearly visible in the figures, as the points farthest away from the query are almost always ranked last. It appears that staircase enforced methods perform better on distance to query while keeping a good distribution. The staircase enforced rankings can be implemented efficiently by adapting the algorithms we presented before.

4 Conclusions

This paper introduced distributed relevance ranking for documents that have two scores. It is particularly useful for geographic information retrieval, where documents have both a textual and a spatial score. The concept can easily be extended to more than two scores, although it is not clear how to obtain efficient algorithms that run in subquadratic time. The experiments indicate that both requirements for a good ranking, distance to query and spreading, can be obtained simultaneously. Especially the staircase enforced methods seem to perform well. User evaluation is needed to discover which ranking method is preferred most, and which parameters should be used.

We have examined more extensions and performed more experiments than were reported in this paper. For example, we also analyzed the case where the unranked points are only related to the 10 (or any number of) most recently ranked points, to guarantee that similar points are sufficiently far apart in the ranked list. Also for this variation, user evaluation is needed to determine the most preferred methods of ranking.

References

Carbonell, J.G., and Goldstein, J., 1998. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Research and Development in Information Retrieval*, pages 335–336.

Cormen, T.H., Leiserson, C.E., and Rivest, R.L., 1990. *Introduction to Algorithms*. MIT Press, Cambridge, MA.

Goldstein, J., Kantrowitz, M., Mittal, V.O., and Carbonell, J.G., 1999. Summarizing text documents: Sentence selection and evaluation metrics. In *Research and Development in Information Retrieval*, pages 121–128.

Goldstein, J., Mittal, V.O., Carbonell, J.G., and Callan, J.P., 2000. Creating and evaluating multi-document sentence extract summaries. In *Proc. CIKM*, pages 165–172.

Harman, D., 2002. Overview of the TREC 2002 novelty track. In *NISI Special Publication 500-251: Proc. 11th Text Retrieval Conference (TREC 2002)*.

Heckbert, P.S., and Garland, M., 1995. Fast polygonal approximation of terrains and height fields. Report CMU-CS-95-181, Carnegie Mellon University.

Jones, C.B., Purves, R., Ruas, A., Sanderson, M., Sester, M., van Kreveld, M.J., and Weibel, R., 2002. Spatial information retrieval and geographi-

cal ontologies – an overview of the SPIRIT project. In *Proc. 25th Annu. Int. Conf. on Research and Development in Information Retrieval (SIGIR 2002)*, pages 387–388.

Rauch, E., Bukatin, M., and Naker, K., 2003. A confidence-based framework for disambiguating geographic terms. In *Proc. Workshop on the Analysis of Geographic References*.
http://www.metacarta.com/kornai/NAACL/WS9/Conf/ws917.pdf.

van Kreveld, M., van Oostrum, R., and Snoeyink, J., 1997. Efficient settlement selection for interactive display. In *Proc. Auto-Carto 13: ACSM/ASPRS Annual Convention Technical Papers*, pages 287–296.

Visser, U., Vögele, T., and Schlieder, C., 2002. Spatio-terminological information retrieval using the BUSTER system. In *Proc. of the EnviroInfo*, pages 93–100.