Comparison of parallel and random approach to a candidate list in the multifeature querying

Peter Gurský

Institute of Computer Science, Faculty of Science**
P.J.Šafárik University Košice
Jesenná 9, 040 01, Košice
gursky@upjs.sk

Abstract. In the field of the multifeature querying it is possible to use many heuristics to retrieve top k objects to became low number of accesses to the sources. When the sources have many equal values, it is often hard to choose which source should be accessed next. In this paper we compare previous random approach with the parallel approach to the set of actual candidate sources.

Key words: multifeature querying, top-k objects, aggregation

1 Introduction

Many times we want to find the best object or top k objects in the possible huge set of objects. The reason of which object is better than the other, is based on the properties of the objects. Such properties are typically fuzzy. For example, when we want to find top k hotels, we can look at a distance from the beach, price per night, number of stars, travel expenses, etc. We need is to specify, how to compute the overall score of each object to became the order of the objects. Moreover all these data can be accessible by different sources (web services).

There are several algorithms in this area, solving this problem. Ronald Fagin introduced "Fagin's algorithm", which solve this problem first time[6]. Fagin et al. [2] presented "threshold" algorithm that made the search much faster. Güntzer et al. [1] defined "quick-combine" algorithm using first heuristics. Other heuristics was presented by P. Gurský and R. Lencses [4].

The "quick-combine" algorithm was originally developed over multimedial data. Such a data are typically continuous e.g. it is very unusual to have two objects with the same value of a property. The experimental comparison of the heuristics [4] showed, that the heuristics used in [1] is quite ineffective, when the sources have few discretized values i.e. number of stars of hotels. In this

** This work was partially supported by the grant VEGA 1/0385/03 and 'Štátna úloha výskumu a vývoja "Nástroje pre získavanie, organizovanie a udržovanie znalostí v prostredí heterogénnych informačných zdrojov" prierezového štátneho programu "Budovanie informačnej spoločnosti".'

paper, we show a possible improvement of the performance of this heuristics over discretized data. We also try to use the same approach to other relevant heuristics presented in [4].

In chapter 2 we describe a formal model of data. In chapter 3 we present a generalized version of all mentioned algorithms and compare three different heuristics. The experimental comparison of the heuristics is showed in chapter 4. Chapter 5 concludes our paper.

2 Model

Assume we have a finite set of objects. Cardinality of this set is N. Every object x has m attributes $x_1, \ldots, x_m$. All objects (or identificators of objects, URI) are in lists $L_1, \ldots, L_m$, each of length N. Objects in list L_i are ordered descending by value of object in attribute x_i. We can define two functions to access objects in lists. Let x be an object. Then $s_i(x)$ is grade (or score, rank) of object x in list L_i and $r_i(j)$ is object in list L_i in j-th position. Lists can be accessed in two ways. First one is $sorted(sequential)access$. Using this type of access the grades of objects are obtained by proceeding through the list sequentially from the top. The second mode of access is $random$[1]$(direct)$ $access$. Here, we request the grade of object x in list L_i and obtain it in one random access.

We have also monotone aggregation function F, which combine grades of object x from lists $L_1, \ldots, L_m$. The overall value of an object x we denote as $S(x)$ and it is computed as $F(s_1(x), \ldots, s_m(x))$.

Our task is to find top k objects with highest overall grades. We also want to minimize time and space. That means we want to use as low sorted and random accesses as possible.

3 Generalized Threshold algorithm and heuristics

For each list L_i, let $u_i = s_i(r_i(z_i))$ be the grade of the attribute of the last object seen under sorted access, where z_i is the number of that possition. Define the threshold value τ to be $F(u_1, \ldots, u_m)$. Because we assume that we have a monotone aggregation function F and the lists are sorted descend by their values, the threshold τ is the value, which none of still unseen objects can reach [2]. Hence when all objects in the top k list have their values greater or equal to the threshold, then this top k list is final and there is none unseen object with greater value. This property is very important to have the algorithm correct.

Let $z = (z_1, \ldots, z_m)$ be a vector, which assigns for each $i = 1, \ldots,$ m the position in list L_i last seen under sorted access. Let H be a heuristics that decides which list (or lists) should be accessed next under sorted access (notice that heuristics can change during the computation). Moreover, assume that H

[1] The term "random access" was firstly used by R. Fagin [2], we keep this original term in spite of overlapping with the term "random approach". The "random approach" to a set means, that we choose one element of this set randomly.

is such, that for all $j \leq m$ we have $H(z_j) = z_j$ or $H(z_j) = z_j+1$ and there is at least one $i \leq m$ such that $H(z_i) = z_i+1$. The set $\{i \leq m : H(z)_i = z_i + 1\}$ we call the set of candidate lists (or simply candidates) for the next sorted access.

In this paper we use three types of heuristics.

First type of heuristics (denote $H1$) do the sorted access in all m lists parallelly. It means, that for each $i \leq m$ holds $H(z_i) = z_i+1$. This heuristics was firstly presented by Fagin et al. [2] in the "Threshold algorithm". This kind of heuristics is used only (if ever) in the first phase of computation to retrieve the beginnings of the lists. Next two heuristics are used in the rest of computation.

The use of the $((\delta F/\delta x) * \Delta x)$ heuristics ($H2$) was first presented by Güntzer et al. [1] as a part of "Quick-combine algorithm". Let us look at the next unequation. To keep algorithm correct, for each object x in the final top k list must hold:

$$S(x) = F(s_1(x), \ldots, s_m(x)) \geq \tau \tag{1}$$

Hence, when we can say, that this unequation holds, we have the final top k list. Obviously, there are two ways to make (1) hold: to increase the left side or to decrease the right side. Heuristics $H2$ tries to decrease τ as fast as possible. As a criterion for a list L_i, $i \leq m$ to be given to the set of candidates for the next sorted access, is to have Δ_i maximal. Δ_i is defined as:

$$\Delta_i = \left(\frac{\delta F}{\delta x_i}(s_1(r_i(z_i)), \ldots, s_m(r_i(z_i)))\right)^{-} * (s_i(r_i(z_i - p)) - s_i(r_i(z_i))) \tag{2}$$

The constant p is some suitable (small) natural number. Hence Δ_i is the multiplication of the partial derivative of aggregation function F from the left in the point $(s_1(r_1(z_1)), ..., s_m(r_m(z_m)))$ and the expected change of values in p steps (Δx factor) of the i-th list. When we have Δ_i for each $i \leq m$, we can set the value of $H(z_i)$. Heuristics $H2$ sets $H(z_i)=z_i+1$ if $\Delta_i = \max\{\Delta_j; j \leq m\}$. Otherwise $H(z_i)=z_i$. The only necessary condition we required from F is the continuity from the left.

The $(\delta F/\delta x) * x)$ heuristics ($H3$) is a variation of the last one. This heuristics was presented by Gurský et al.[4] at the first time. Instead of the Δx factor, $H3$ chooses an x-factor, thus the last seen value in the i-th list. The criterion for this heuristics is:

$$\chi_i = \left(\frac{\delta F}{\delta x_i}(s_1(r_i(z_i)), \ldots, s_m(r_i(z_i)))\right)^{-} * s_i(r_i(z_i)) \tag{3}$$

The criterion (3) computes the partial derivation of F from the left in the point $(s_1(r_1(z_1)), ..., s_m(r_m(z_m)))$ and multiply it with value in the point $s_i(r_i(z_i))$ in L_i. This heuristics sets $H(z_i)=z_i+1$ if $\chi_i = \max\{\chi_j; j \leq m\}$. Otherwise $H(z_i)=z_i$. We need the continuity from the left for the function F again.

When the aggregation function is the simple weighted mean, the derivation used by these heuristics is a constant, more precise it is the weight of the attribute.

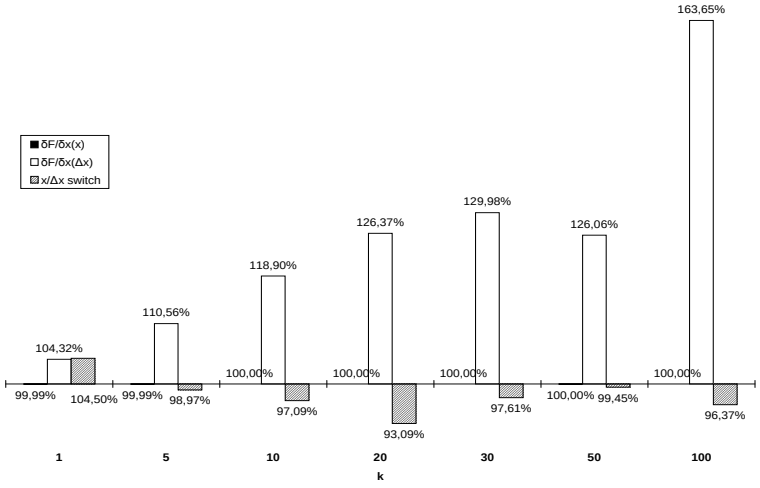

Fig. 1. Change of performance using parallel approach (benchmark data)

Now we can describe the generalized Threshold algorithm:

0. z:=(0,...,0), in case of algorithms $((\delta F/\delta x) * \Delta x)$ or $\Delta x/x$, set the suitable small natural p.
1. Set the heuristics H:
   - $\overline{((\delta F/\delta x) * x)}$: H:=$H3$
   - $\overline{((\delta F/\delta x) * \Delta x)}$: if any $z_i < p$ then H:=$H1$; otherwise H:=$H2$
   - $\overline{\Delta x/x}$: if any $z_i < p$ then H:=$H1$; otherwise if H=$H1$ then H:=$H2$; if H=$H2$ then H:=$H3$; and if H=$H3$ then H:=$H2$.
2. – **parallel approach**: Do the sorted access in parallel to each of the sorted lists to all positions where $H(z)_i = z_i$+1. Put $z_i = H(z)_i$.
   – **random approach**: Do the sorted access to randomly chosen sorted list L_i where $H(z)_i = z_i$+1. Put $z_i = z_i + 1$ and for each $j \leq m, j \neq i$ do nothing.
3. First control: Compute the threshold value τ. As soon as at least k objects have been seen whose grade is at least equal to τ , then go to step 6.
4. For every object x that was seen under sorted access in the step 2, do the random access to the other lists to find the grade $s_i(x)$ of object in every list. Then compute the grade $S(x) = F(s_1(x), \ldots, s_m(x))$ of object x. If this grade is one of the k highest ones we have seen, then remember object x and its grade $S(x)$.
5. Second control: As soon as at least k objects have been seen whose grade is at least equal to τ , then go to step 6, otherwise go to step 1.

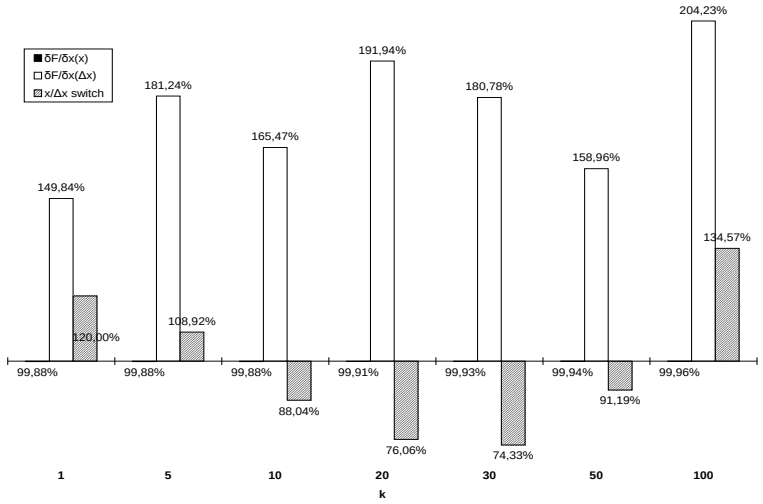

Fig. 2. Change of performance using parallel approach (artificial data)

6. Let Y be a set containing the k objects that have been seen with the highest grades. The output is then the graded set $\{(x, S(x)) : x \in Y\}$.

Individual algorithms differ in the steps 2 and 3. Shortly the $(\delta F/\delta x)*x)$ algorithm use heuristics $H3$ directly on beginning. The Quick-combine (or $((\delta F/\delta x)* \Delta x))$ algorithm use for the first p steps in each list the heuristics $H1$ and than the heuristics $H2$. The $\Delta x/x$ algorithm use for first p steps in each list the heuristics $H1$, too. After that it switches between the heuristics $H2$ and $H3$.

Original algorithms choose the random approach in the step 3. All mentioned algorithms can have their "parallel variant" too. In our experiments we compare all 6 possible variants.

4 Experiments

4.1 Testing data

Benchmark data The first sort of data are real data that come from randomly generated queries in information retrieval system with support of relational database. We used different combinations of local and global weights as different ways for weighting of occurrences terms in documents to generate 6 sets of benchmark data. Each set include over 25 000 objects (documents) with 50 different attributes (terms). To measure all particular experimental results we compare the average values from 6 sets of these benchmark data with the use of aggregation functions with randomly chosen weights for each list. Histograms of

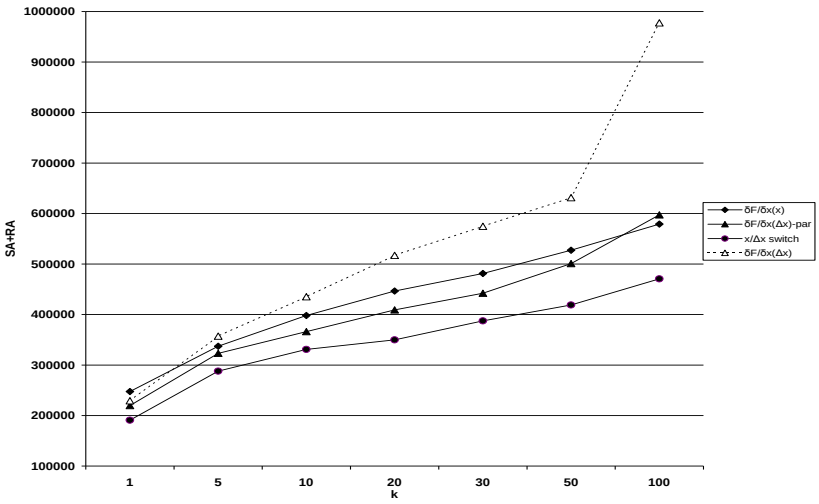

Fig. 3. Number of all accesses (benchmark data)

data are exponential - there is very low number of objects with high values and a lot of objects with low values. Such distribution is typical in the information retrieval area but in many other areas too.

Artificial data The second sort of data were generated with various distribution of values. We used 2 exponential and 2 logarithmic distributions with 10000 objects and 6 types of aggregation functions. The values of the attributes was rounded to 10 discrete values. Such a discretisation is quite common in real data i.e. number of stars of hotels, rating classes of companies, and other human produced ratings. Finest discretisation can be i.e. prices of some product or a guess for the length of a trip. Continuous data are typical for some physical experiments, precise measurements or multimedial data. In this paper we focus on the discrete data. Using different combination of the source data and aggregation functions we became 16 different inputs for algorithms. In the final results we use the averages of the particular results.

4.2 Results

In our experiments we wanted to compare the random and parallel approach to the set of candidates. On the figure 1 and 2 we take as the base the random approach (=100%). We can see the performance of the parallel approach compared with the random approach for each heuristic. Both artificial and benchmark data shows, that different approach to the candidates does not change anything for

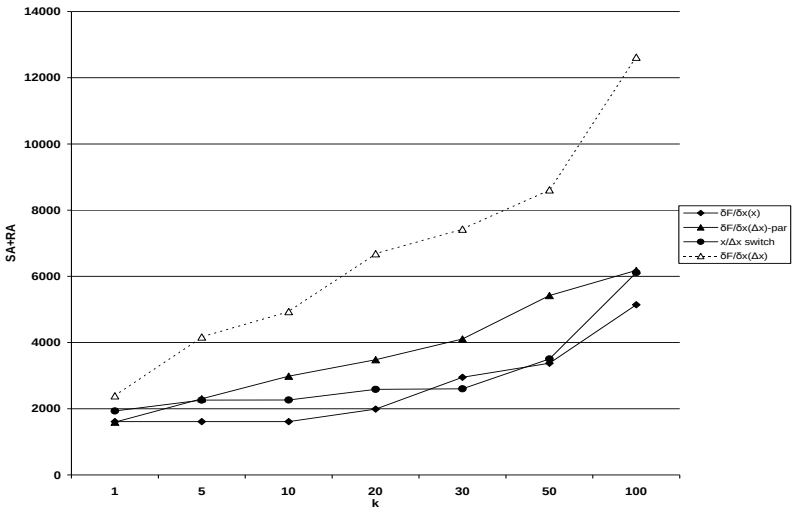

Fig. 4. Number of all accesses (artificial data)

the $(\delta F/\delta x) * x$ heuristics. For the $x/\Delta x$ heuristics the random approach seems to be better in most cases.

In our results, the new parallel approach positively improved the performance of the $(\delta F/\delta x) * \Delta x$ heuristics over discretised values. This heuristics was originally developed for multimedial data. Using our improvement this heuristics can be quite efficient over discretized data too, but as we can see in figures 3 and 4 when we work over discretized values the new approach to the $(\delta F/\delta x) * \Delta x$ heuristics is still not better than quite stable $x/\Delta x$ heuristics.

Why we have such a results? The parallel version of the $(\delta F/\delta x) * x$ algorithm has almost same results as its random variant maybe because there was almost none situation in which we had more than one candidate in the candidate set. On the other side the $(\delta F/\delta x) * \Delta x$ algorithm had many such a situations. The reason of this situation is the fact, that the expression $(s_i(r_i(z_i - p)) - s_i(r_i(z_i)))$ almost always equals to zero for small p and discrete values in the lists. As experiments show, for this heuristics when we don't know to prefer one list, it is better to access all lists. For the $x/\Delta x$ heuristics seem the random approach to be better when k is greater or equals to 5.

5 Conclusions

We proposed the new parallel approach to the candidate set and compare it with the previous random approach. We experimentally showed that this type of approach improved the $(\delta F/\delta x) * \Delta x$ heuristics over discrete data. On the

other hand the $x/\Delta x$ heuristics keeps its first place in lower number of accesses as was shown in [4].

References

1. U.Güntzer, W.Balke, W.Kiessling *Optimizing Multi-Feature Queries for Image Databases*, proceedings of the 26th VLDB Conference, Cairo, Egypt, 2000
2. R.Fagin *Combining fuzzy information from multiple systems*, J. Comput. System Sci., 58:83-99, 1999
3. R.Fagin, A.Lotem, M.Naor *Optimal Aggregation Algorithms for Middleware*, proc. 20th ACM Symposium on Principles of Database Systems, pages 102-113, 2001
4. P.Gurský, R.Lencses *Aspects of integration of ranked distributed data*, proc. Datakon, 2004
5. R.Fagin *Combining Fuzzy Information: an Overview*, ACM SIGMOID Record 31, Database principles column, pages 109-118, 2002
6. R.Fagin *Combining fuzzy information from multiple systems*, 15th ACM Symposium on Principles of Databases Systems, pages 216-226, 1996
7. P.Gurský, R.Lencses, P.Vojtáš *Algorithms for user dependent integration of ranked distributed information*, technical report, 2004