Performance and Quality Evaluation of a Personalized Route Planning System

Wolf-Tilo Balke[1], Werner Kießling[2], Christoph Unbehend[2]

[1]Electrical Engineering and Computer Science, University of California, Berkeley, USA
balke@eecs.berkeley.edu
[2]Institute of Computer Science, University of Augsburg, Germany
{kiessling, unbehend}@informatik.uni-augsburg.de

Abstract

Advanced personalization of database applications is a big challenge, in particular for distributed mobile environments. We present several new results from a prototype of a route planning system. We demonstrate how to combine qualitative and quantitative preferences gained from situational aspects and from personal user preferences. For performance studies we analyze the runtime efficiency of the SR-Combine algorithm used to evaluate top-k queries. By determining the cost-ratio of random to sorted accesses SR-Combine can automatically tune its performance within the given system architecture. Top-k queries are generated by mapping linguistic variables to numerical weightings. Moreover, we analyze the quality of the query results by several test series, systematically varying the mappings of the linguistic variables. We report interesting insights into this rather under-researched important topic. More investigations, incorporating also cognitive issues, need to be conducted in the future.

1. Introduction

Given the overwhelming amount of information accessible over the Internet user preferences and personalization have become an essential necessity for information systems. Personalized search engines for E-commerce are one well-known application domain (e.g. [19], [15]). The demand for personalized query answering is obvious for other important application areas like traffic information systems supporting individual route planning. Whether a certain route is better than another route critically depends on situational preferences and on personal user preferences. Recently preference modeling and preference query languages have gained substantial attention. E.g. [7], [14] propose powerful and efficient ways to deal with preferences. We will adopt the approach of [14], providing a unified model for qualitative and quantitative preferences. Preferences are treated as soft constraints with an intuitive "I like x better than y" semantics. For instance, most people like a dry road better than a wet and slippery one. However, this judgment in general depends on the various circumstances, e.g. whether the person drives a motorbike, a roadster or a family car. Mathematically [14] characterizes such preferences by strict partial orders. Since many preference decisions for route planning involve both numerical data and qualitative aspects, a good combination of both forms of preference modeling is essential. In [17] we showed how this can be achieved for the domain of document retrieval, fostering the expressiveness of preference-based retrieval systems.

Applying this powerful framework to a mobile traffic information system with real-time constraints, in [3] we investigated the use of numerical preferences. We implemented an architecture featuring the delivery of personalized route planning to a variety of mobile devices. This is done on the basis of current traffic information that is dynamically gathered from several Internet sources. The intention is that given current information about possible routes, traffic jams, construction sites, weather conditions, etc., our system calculates a certain number of routes that would serve the user's preferences best. For the efficient real-time evaluation of numerical preferences we use the novel SR-Combine algorithm [2]. A system prototype has been demonstrated recently at [4].

The innovative contributions of this paper will be twofold. Firstly, more interesting performance evaluation results for SR-Combine will be presented. Secondly, we investigate the crucial quality issue for personalized route planning. Despite its obvious importance this topic of quality assessment has not gained much attention by database researchers in the past. In particular we will focus on intuitive preference modeling using linguistic variables to express a user's notion of importance for each of a route's characteristics. Since a direct assignment of numerical weights can hardly be intuitive (e.g. what does a weight of 0.37 for avoiding traffic jams mean?) the concept of a linguistic variable, proposed already by [20], will be explored. Thus within the course of this paper we will show

- how to sensibly extend numerical preferences with qualitative preferences to improve the results on our personalized route planning, and
- how different mappings of linguistic variables to numerical weights used for numerical preferences will affect the quality of the query results.

This rest of the paper is organized as follows. To be self-contained section 2 revisits the preference framework and its application to our personalized route planning system. Section 3 reports efficiency evaluation results of numerical preferences using the SR-Combine algorithm. In section 4 we deal with the effects of linguistic variables and their impacts on the quality of preference query results. Related work is discussed in section 5. We will conclude with a short summary and outlook in section 6.

2. Preference Engineering for Personalized Route Planning

We shortly review those parts of the preference framework from [14] needed here.

2.1. Preferences Constructors

Given a set A of attributes, a *preference* P is denoted as **P = (A, $<_P$)**, where $<_P \subseteq$ dom(A) $\times$ dom(A) a strict partial order and x $<_P$ y expresses that "y is better than x". Preferences can be engineered inductively by a variety of intuitive *preference constructors,* being distinguished between *base constructors* and *complex constructors*.

For the scope of this paper we require the following base preference constructors:

- Given a *scoring function* f: dom(A) $\rightarrow$ [0,1], then P is called **SCORE preference** (denoted as **P := SCORE(A, f)**), if for all x, y $\in$ dom(A): x $<_P$ y iff f(x) < f(y)

- Given *NEG-set* $\subseteq$ dom(A), P is called **NEG preference** (denoted as **P := NEG(A, NEG-set)**)), if for all x, y $\in$ dom(A): $x <_P y$ iff $x \in$ NEG-set $\wedge$ $y \notin$ NEG-set.

On top of the base preferences the following complex preference constructors are required for the scope of this paper:

- Given preferences $P_1 = (A_1, <_{P1})$ and $P_2 = (A_2, <_{P2})$, then $P = (\{A_1, A_2\}, <_P)$ is called **prioritized preference** (denoted as **$P := P_1$ & P_2**), if for all $x = (x_1, x_2)$, $y = (y_1, y_2) \in \mathrm{dom}(A_1) \times \mathrm{dom}(A_2)$: $x <_P y$ iff $x_1 <_{P1} y_1 \vee (x_1 = y_1 \wedge x_2 <_{P2} y_2)$
- Given preferences P_1 := SCORE(A_1, f_1), P_2 := SCORE(A_2, f_2) and a monotonic *combining function* F: $[0,1] \times [0,1] \to [0,1]$, then $P = (\{A_1, A_2\}, <_P)$ is called **numerical preference** (denoted as **P := rank$_F$(P_1, P_2)**)), if for all $x = (x_1, x_2)$, $y = (y_1, y_2) \in \mathrm{dom}(A_1) \times \mathrm{dom}(A_2)$: $x <_P y$ iff $F(f_1(x_1), f_2(x_2)) < F(f_1(y_1), f_2(y_2))$. The generalization to n preferences is obvious.

Summarizing the effects of these preference constructors, SCORE prefers values with higher f-scores, NEG prefers values not in the NEG-list, prioritization '&' treats preference P_1 as more important than P_2, and numerical 'rank$_F$' prefers values with higher combined F-values.

2.2. Situational Preferences for Route Planning

Let's take a closer look at the preferences involved in route planning. Relevant attributes for route planning include (but are not restricted to) the length of a route, the number and length of traffic jams and road work sites, and the weather condition. All these information is gathered from distributed Internet sources and assembled using numerical preferences, since it is a reasonable assumption that the different characteristics should be able to compensate each other. For instance a user might be interested to take a little longer route, if it is free of traffic jams. Thus compromises can in most cases be reached. We have engineered six *situational* preferences P_1, …, P_6 within our traffic information scenario (see [3] for more details):

- **P_1 := SCORE(length, f_1):** The f_1-score of routes is estimated from their lengths.
- **P_2 := SCORE(traffic_jams, f_2):** The f_2-score is influenced by information about the number of jams and their total lengths. We also distinguish between two types of jams: standstill or stop-and-go traffic. Standstill jams are assigned double lengths.
- **P_3 := SCORE(road_works, f_3):** Basically P_3 works analogous to P_2. Here we distinguished between road works with closed lanes and those without.
- **P_4 := rank$_{F4}$(P_{4a}, P_{4b}, P_{4c}, P_{4d}):** Weather information is a complex numerical preference of 4 SCORE preferences on average temperatures, rainfall, wind and humidity:
  - P_{4a} := SCORE (avg_temp, f_{4a}), P_{4b} := SCORE(avg_rainfall, f_{4b}),
  - P_{4c} := SCORE(avg_wind, f_{4c}), P_{4d} := SCORE(avg_humid, f_{4d})

  The combining function F_4 models typical traffic situations, e.g. that the dangers of black ice caused by some combinations of temperature and rainfall are more important than wind and humidity. In our prototype we used
  $F_4(x) := (2 \cdot f_{4a}(\text{avg\_temp}) + 2 \cdot f_{4b}(\text{avg\_rainfall}) + f_{4c}(\text{avg\_wind}) + f_{4d}(\text{avg\_humid})) / 6$
- **P_5 := SCORE(road_quality, f_5):** The f_5-score assesses what percentage of each route can be driven on motorways, state highways and general roads. For each category a

weight is assumed: Since it is most convenient to drive a possibly high percentage on fast motorways, they are assigned the highest weight, state highways are assigned a medium weight and general roads are assigned the lowest weight.

- **P_6 = SCORE(expected_jam, f_6):** The f_6-score reflects the probability that traffic jams might occur on each specific route. It is derived from statistical information. Routes with frequent jams are assigned lesser scores than routes where jams occur seldom or even have occurred not at all.

Given these situational preferences an overall numerical preference P for route planning is constructed, now incorporating further *individual* user preferences. However, at this point where individual human judgment is involved, instead of a numerical weighting more intuitive *linguistic variables* are used to specify the combining function F. The user may distinguish five degrees of importance (*unimportant, normal, medium, important, very important*) that are automatically mapped onto certain weights w_i for F. (The crucial question, what is a proper choice for this mapping, will be discussed in deep in section 4.)

- **P := rank$_F$(P_1, P_2, P_3, P_4, P_5, P_6):** For a route x = (length, traffic_jams, road_works, (y_1, …, y_4), road_quality, expected_jam) the combining function F is defined as:
  $F(x) = ((w_1+1)\cdot f_1(\text{length}) + w_2\cdot f_2(\text{traffic\_jams}) + w_3\cdot f_3(\text{road\_works}) + w_4\cdot F_4(y_1, \ldots, y_4)$
  $+ w_5\cdot f_5(\text{street\_quality}) + w_6\cdot f_6(\text{expected\_jam})) / (S\ x_i + 1)$

2.3. Qualitative Preferences for Avoiding Certain Situations

Though our overall preference P aims to deliver the best route for each user, the way in which characteristics can be compensated sometimes leads to wrong results. If for instance a road should be entirely closed due to an accident, it is of course free of jams behind the blocking, may also show no road works and the weather may be fine. Thus it will not be enough to set one or even a couple of the base preferences to the worst value, because other preferences could compensate these preferences and though entirely blocked, the respective route may nevertheless end up with an overall agreeable score. Extending the preference engineering reported in [3], we now show how to integrate numerical and qualitative preferences.

Employing NEG preference constructors as defined before we can elegantly fix this behavior. E.g., for the case of road blockings with an attribute *blocked* or for the case of black ice with *black_ice* we can construct two suitable NEG preferences as follows:

- P_{block} := NEG (*blocked*, {true}), i.e. blocked routes are avoided if available.
- P_{ice} := NEG(*black_ice*, {true}), i.e. routes with black ice are avoided if possible.

Inductive complex preference construction now enables the integration of these qualitative preferences with our previous numerical preference P. Since our intention is to avoid all blocked roads if possible, preference P_{block} is most important, i.e. it is *prioritized*. But we also want to avoid black ice on streets, hence preference P_{ice} becomes important in second place. After we have tried to avoid blocked or dangerous routes, our previous numerical preference P gets its turn. Thus as an improved overall preference we get:

- $P_{final} := P_{block}\ \&\ P_{ice}\ \&\ P$

Please note that a hard constraint in an SQL WHERE-clause would generally exclude icy roads, even if there is no other choice. As our intention is not to patronize users whether to drive in certain weather conditions, using the prioritization is the right choice: Instead of committing the notorious *empty-result bug*, the user gets a choice. Since preferences are *soft* constraints we can still deliver the best possible routes, even if all routes that could be used should be icy. This implies that the final decision, whether to take the risk of driving under these circumstances, can only be taken by the user. Thus preference-based route planning is cooperative, offering a high degree of ease of use and flexibility for the user.

3. Performance Evaluation of Numerical Preference Queries

Our mobile route planning system has to collect information in the form of sorted score lists from different Internet sources and has to combine them efficiently into $P_{final} := P_{block}$ & P_{ice} & P. In the sequel we will report further performance results for evaluating a preference query, given a numerical preference like $P := rank_F(P_1, P_2, P_3, P_4, P_5, P_6)$.

3.1. Prototype Architecture for Mobile Route Planning

To build an efficient preference-based mobile route planning system we need an architecture capable of transforming service requests from a variety of mobile clients. The SR-Combine [2] algorithm runs over a database that holds all up-to-date information and delivers best matching results to the respective client device. Figure 1 shows the architecture of our prototype. The service is built around an IBM DB2 V7.2 database system storing all different routes together with geographic, traffic and weather information. For the computation of routes only parts between adjacent cities are stored in tables. For queries on longer routes the transitive closure is built on demand from these smaller parts. This leads to efficient storage of routes, because we do not need to store over 100000 routes (given our sample data covering a part of the German Ruhrgebiet), but all sensible routes for each query are deduced with the restriction that no route is longer than 1.5 times the shortest route and no cycles appear.

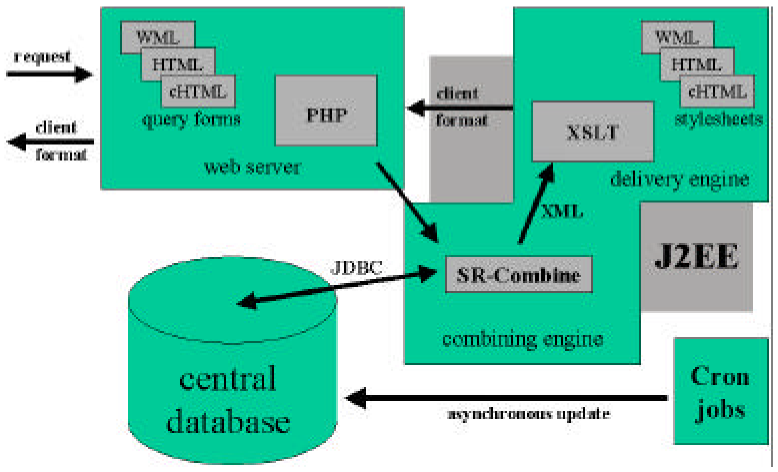


Figure 1: Architecture of a mobile preference-based route planning system

Since tests on multi-feature combination systems using on-line sources, e.g. for restaurant booking like [6], show answer times in the area of minutes to hours to complete more complex queries, we decided for a central data storage, being updated in suitable intervals depending on the kind of information. Our central database server is updated regularly in an asynchronous manner using the WSQ/DSQ approach of [10]. In our UNIX system environments we used a set of cron-jobs that supply the database with updates according to the update profile of the data source. For instance, traffic jam information is far more frequently updated than road work information which tends to be more durable. In our current implementation we use update intervals of 1 minute updates for traffic information, 30 minute updates for weather information and 6 hour updates for road works. On top of the database our prototypical architecture uses an application server that communicates via a Web server. At each service request we identify the browser type of the mobile client device and hand on the form data to our application server running the combining engine with the SR-Combine algorithm. When the final result is assembled, the combining engine uses a generic XML format that can be transformed by the delivery engine using XSLT to suit any registered client device. For each registered device an appropriate style-sheet is kept by the delivery engine.

3.2. Performance Results Using SR-Combine

A central part for the evaluation of numerical preferences is the query engine. The problem of efficiently integrating several ranked lists to get some overall best objects has been addressed as *top-k retrieval*, e.g. [9], [11]. We have implemented the SR-Combine algorithm [2] that promises fast top-k retrieval for central server architectures, but as a unique feature also allows to be smoothly migrated to distributed architectures, if high bandwidths and high performance networks can be guaranteed eventually.

SR-Combine intelligently self-adapts to a variety of environments and controlling runtime costs. For each source a *cost ratio* is assigned, determining the relative costs for different kinds of accesses to that source. SR-Combine can choose between accesses iterating a ranked list (*sorted access*) and accessing the score of a certain object directly (*random access*). By carefully balancing the access costs and the expected use of the access SR-Combine optimizes its total run-time. Benchmarks in [2] show that SR-Combine scales well and in many cases beats the psychologically critical 3-second response-time threshold. Our Java-implementation of SR-Combine basically works as follows: Sorted accesses are implemented by opening a JDBC result set on the application server with a query posed to our central database, whereas the random accesses are more expensive and rely on suitable indexes for each preference.

Due to the necessary updates in short time intervals, no updated preference scores are directly written into the database, but are cached and written asynchronously. For improved efficiency the application server has transparent access to these caches to execute preference queries. To avoid update problems in materialized results for each query new JDBC result sets are created and closed after the completion of the request. New requests can than be performed over the possibly refreshed cached data. For the generation of JDBC result sets and subsequent sorted accesses we measured an average time of *1.56 msec per sorted access*. Using the caches and employing suitable cluster indexes the time for random accesses has been improved to *25 msec per random access* on average. Thus the *cost-ratio* is *1:16* here.[1] Note that the accurate knowledge of this cost ratios is crucial in tuning the performance of SR-Combine. In figure 2 one clearly see that expectedly the average run-time increases with more objects to return. But by knowing the right cost-ratio SR-Combine automatically adapts its control flow to improve its performance in the given configuration.

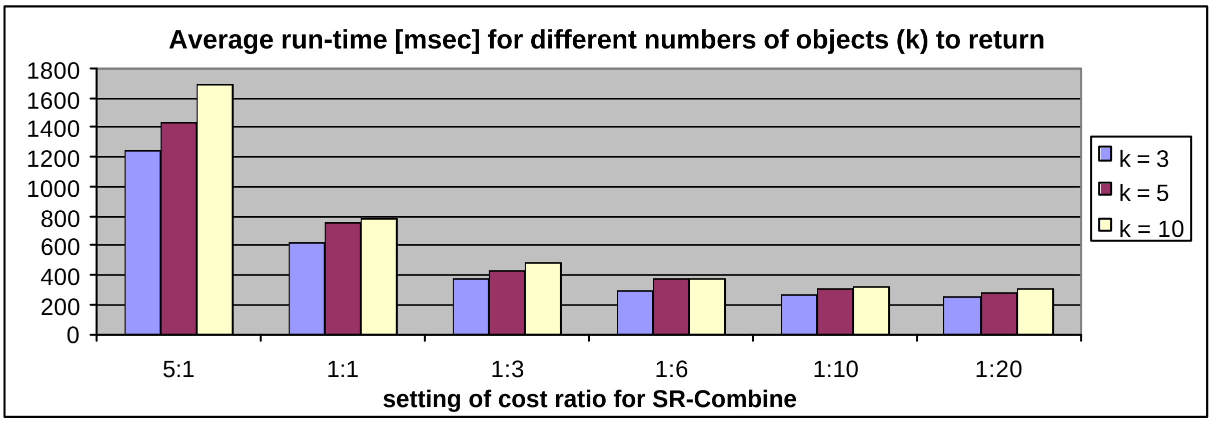


Figure 2: Runtimes of SR-Combine with different cost ratios.

4. Quality Evaluation of Numerical Preference Queries

The performance evaluation of our personalized route planning service has been done for several use cases in [3]. But using a technique like numerical preferences, we also have to investigate the question of how to assess the quality of the result sets gained. The semantic meaning of user-specified weightings in preferences is a problematic issue, requiring serious investigations. What does it mean, if an aspect is two times more important than any other aspect and how do we decide, if it is two times or three times or just 1.5 times? In contrast to top-k retrieval, fuzzy systems addressed the notion of linguistic variables already as early as 1975 [20]. We will utilize such linguistic variables and investigate their effect on the result sets.

4.1 Experimental Setup and Geometrical Model

We focused on the six different aspects of routes discussed in section 2. Since our overall preference integrates them into a weighted sum preference, we have to assign six user-determined weights, representing their personalized importance. We experimented with different settings and eventually decided on five linguistic terms that are mapped onto different weightings. We considered two different sets of weightings that are identical in dropping preferences by assigning a value of 0 and taking 1 as the default weight. The definition of importance is the done by either using equidistant steps or using rounded doubled weighting for an increase of importance, thus putting more emphasis on growing importance:

[1] This cost-ratio for sorted to random accesses can considerably differ from application to application, depending on the specific architecture, average workloads, update rates and network latencies. For instance in [18] a system for on-line auctions has been presented showing a cost ratio of only 1:6.

- *'unimportant'* assigns a weighting of 0
- *'normal'* defines the default weighting of 1
- *'medium'* defines a weighting of 2 for the equidistant case and 3 for the other setting
- *'important'* defines a weighting of 3 (equidistant) or 5
- *'very important'* defines a weighting of 4 (equidistant) or 10

Note that due to a normalization in the combining function these weightings are invariant to linear scaling. Essential is the ratio between the numeric representations of each set of values.

We can give a useful *geometrical model* that shows how the choice of routes is influenced, if the weightings are altered. Consider an n-dimensional Euclidian space where each dimension is connected to one of the aspects. If we take the cube that is given by the interval between 0 and 1 on each axis, we can represent each route as a point in the cube by assigning its respective scores in an aspect to the respective dimension. Since we have weighted sum numeric preferences, the best points will be situated near a corner of the cube showing score values near 1 in each dimension. Beginning from the point (1, 1, …, 1) we can now collect the best possible routes using a hyper-plane sweeping the volume of the cube. The user-specified weighting tilts this hyper-plane in favor of the higher weighted dimensions. Thus depending on the weighting different objects will be considered for the result set. For the ease of understanding we show this model and the influence of weightings in the case of two dimensions.

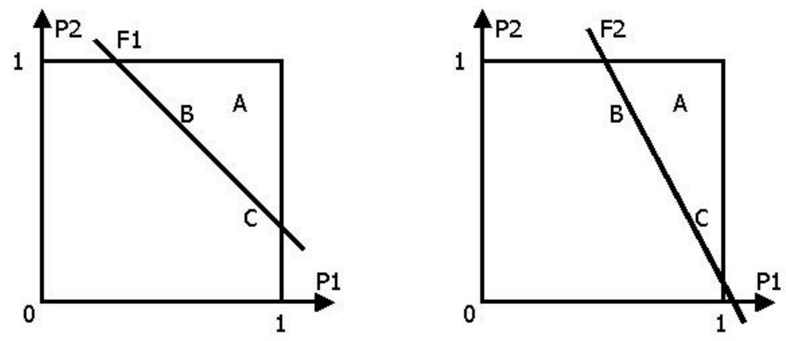


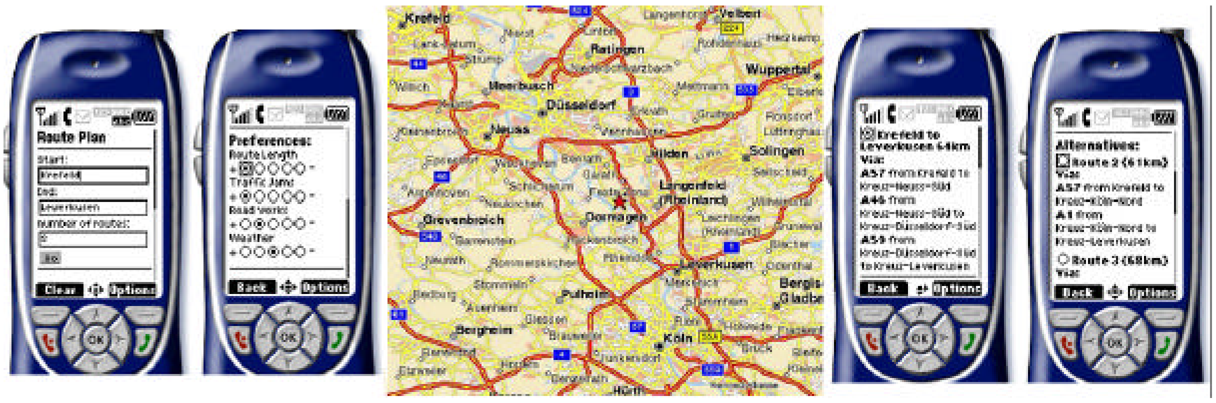
Figure 4: Numeric preference over two base preferences using different combining functions

Consider the example in figure 4. Given two base preferences P_1 and P_2, three different routes A = (0.9, 0.9), B = (0.8, 0.85) and C = (0.95, 0.5) and a numeric preference $\text{rank}_{F1}(P_1, P_2)$ with F_1 as the equally weighted sum $F_1 = (P_1 + P_2)/2$, we experience the situation on the left hand side. When trying to get the best objects, we move the hyper-plane F_1 from point (1, 1) towards point (0, 0) and collect all the objects that are swept by the hyper-plane. In our small example this would be A with an aggregated score of 0.9, B with 0.825 and C with 0.725. If a user should choose to put more importance onto preference P_1 and sets it from 'normal' to 'important' the weighted sum will change to $F_2 = (3*P_1 + P_2)/4$ and the aggregated scores change to A with 0.9, B with 0.8125 and C with 0.8375. This behavior is shown in the right hand side of figure 4. Please note that now B is no longer the second best object but has changed its place with route C due to the tilt in the hyper-plane F_2.

The effect of routes changing places, however, is sensible. Assuming two users, one driving a car and the other driving a motorcycle, let’s consider a weather preference P_1 and a length preference P_2. Clearly route A is always a good idea because it shows a very high consistency of performance of good weather with a short route. Route B shows worse weather and only a slightly longer route, whereas route C shows good weather but a rather long route. For the car driver a little bit of rain will not account for choosing a longer route, hence routes A and B are good choices. On the other hand, a motorcyclist might try to avoid getting wet in the rain, even if that means taking a longer route, hence changing his/her preferences accordingly to an emphasis on P_1. Again route A is a good choice due to the consistency of performance, however, the longer but dry route C is in this case favored over route B.

4.2 Experimental Results

Understanding this behavior we investigated the general stability of the result set. We carried out separate experiments, one mapping our five linguistic terms to equidistant weights (0, 1, 2, 3, 4), the other mapping them to (0, 1, 3, 5, 10). In our route planning prototype a mobile user is enabled to check boxes representing the importance of a specific feature.


Route Plan
Start:
Krefeld
End:
Leverkusen
number of routes:
3
Go
Clear Options

Preferences:
Route Length
+ ◉ ○ ○ ○ ○ -
Traffic Jams
+ ◉ ○ ○ ○ ○ -
Road works
+ ○ ◉ ○ ○ ○ -
Weather
+ ○ ○ ◉ ○ ○ -
Back Options

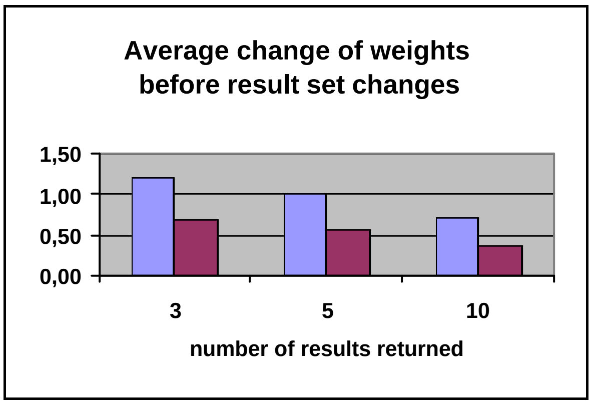
Krefeld to Leverkusen 64km
Via:
A57 from Krefeld to Kreuz-Neuss-Süd
A46 from Kreuz-Neuss-Süd to Kreuz-Düsseldorf-Süd
A59 from Kreuz-Düsseldorf-Süd to Kreuz-Leverkusen
Back Options

Alternatives:
Route 2 (61km)
Via:
A57 from Krefeld to Kreuz-Köln-Nord
A1 from Kreuz-Köln-Nord to Kreuz-Leverkusen
○ Route 3 (68km)
Via:
Back Options

Figure 5: Querying, assessing preferences and delivering the result for mobile route planning

Then the system automatically assigns the appropriate numerical weights and assembles the respective combining function, as illustrated in figure 5. Given the road net (middle) a query can be posed and preferences assessed on all important aspects (left). After the retrieval with SR-Combine the best matching result and some alternative routes are delivered (right).

Mapping linguistic variables onto the chosen numerical values (0, 1, 2, 3, 4) or (0, 1, 3, 5, 10) we have to guarantee that these values are sensible, i.e. that minimal deviations will not immediately affect the result set, leading to a chaotic behavior when personalizing. To this end we carried out a test series with 3000 cases, each time randomly choosing a route and a set of linguistic terms. Considering the such defined combining function we changed a random set of weightings in small steps of 0.01 until the result set changed. Taking the averages we were subsequently able to determine the length of the interval (i.e. the average increase of weights) until the result set changed.



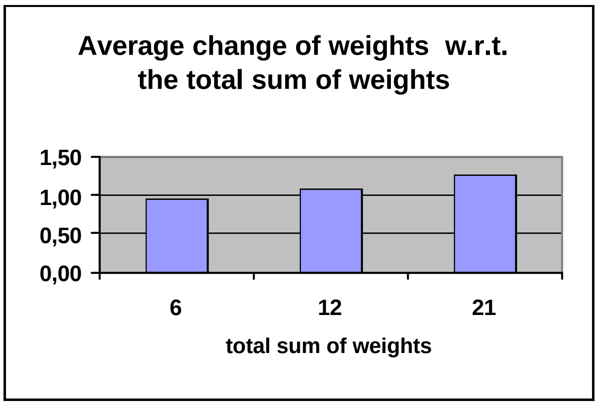


Figure 6: Stability of result sets changing the weights and the total sum of weights

Figure 6 on the left-hand side shows a diagram of the average interval lengths for result sets of 3, 5 and 10 objects to be returned to the user. The mapping onto (0,1,2,3,4) is shown in the dark columns and the mapping onto (0,1,3,5,10) in the light columns. The non-equidistant case in all instances clearly shows a higher average stability; its stability intervals are about double in size. As we can see for our choice of non-equidistant mapping we can change weights about +/- 1 without affecting the result set in the top objects. For instance we could (in the average case) for the top 5 objects to return instead of using the set (0, 1, 3, 5, 10) also have shifted the weightings to (0, 1, 4, 6, 9) or (0, 1, 2, 5, 11) without affecting the top 5 objects. As would be expected intuitively, the result set gets more sensitive towards changes in the weights the more objects it contains. That means that though the top objects are more stable to changes (changes of weights of about 1.3), worse objects can be changed to similar objects more easily (changes of weights of about 0.8). Please note that all the results gained here are averages over a large number of cases. Instances with unusual clusters of data might be far more sensitive even to small changes in the weightings.

On the right hand side of figure 6 the diagram shows how the stability is affected, if the total sum of weights is increased (here we use the non-equidistant mapping). It shows the case of 5 objects to return when the sum of weight is 6 (all streams weighted ‘normal’) , 12 (one stream is weighted ‘important’, one with ‘medium’ and four ‘normal’) or 21 (one ‘very important’, one ‘important’, one ‘medium’ and three ‘normal’). Using the same set of data for each case again the average change of weights before the result set changes is determined. As can be easily seen, the higher the total sum of weights is, i.e. the more dominant single aspects are weighted, the more robust the result set becomes against changes. Thus our mapping of linguistic variables onto the numerical values chosen seems to be sensible. Of course for every system this balance of aspect’s score values and numerical weights may change and has to be considered with respect to its stability.

With respect to the quality we can state that our set of weightings with higher distances between weights will generally show a more stable behavior, because on one hand the total sum of weightings can be expected to be larger, automatically leading to a higher stability (cf. Figure 6, right-hand side). On the other hand, increased differences in the weightings put an strong emphasis on only some of the underlying score preferences. Of course the absolute change of +/- 2 given a weighting of height 10 is showing less effect than the same change on a weighting of 4. Even if the scores are normalized in the sum, the relative effect on each of the underlying scores stays the same. Thus we could produce highest stabilities by simply putting very high weightings on some aspects which would show a similar effect than prioritizing this aspect. Since this behavior is obviously not wanted, our stability measure here can give us only a flavor that a specific set of weightings in the combining function performs reasonably well. A good quality measure in this case hence cannot be only of an analytical nature, but has to involve some practical test cases. Only experiences when working with the systems for real world traffic planning allow for an eventual fine-tuning of weightings by the user. Collected statistics can further help to determine how e.g. the relation between the quality of a road and their jamming behavior is, or by how much traffic is generally slowed down by adverse weather conditions. For instance each user might have a feeling for the danger of foggy conditions, but how much visibility really will lead to an increased risk of accident is beyond the grasp of a mere analytical quality measure and topic to empirical studies. Thus in cooperation with the user-specified relative importance, also averages of practical tests have to be incorporated for real-world systems.

5. Related Work

Efficiency of top-k retrieval can be distinguished in middleware and database algorithms. Middleware algorithms, e.g. [9], [11], focus on the cost of object accesses and employ techniques for accesses that can be executed over networks on distributed data sources. Database algorithms are designed to work with central multi-dimensional indexes, e.g. [5]. Though the latter algorithms offer a better performance by using local object accesses and not via networks with higher latencies, benchmarks in [3] show that the SR-Combine algorithm for the distributed case can often provide a response time below 3 seconds. Though being slower than central index approaches, the possibility of simple migration to distributed architectures and easy inclusion of new content sources make up for a barely noticeable deficiency in performance. Here we have shown the significance of knowing the important cost-ratio of random to sorted accesses. Another approach uses materialized views, e.g. [12]. However, to offer the best possible performance such systems have to materialize all different views that users might want to query, hence being very storage-intensive.

Numerical preferences are rather popular for personalization, see e.g. [1]. Nevertheless, to our knowledge there have been virtually no experiments about their semantic meaning in the database community. In the field of query relaxation a survey for different functions assessing the quality of the result objects by measuring the necessary relaxation steps has been given by [18]. The deepest investigation on the evaluation of preferences by using linguistic variables goes back to fuzzy systems and soft computing [20]. These systems assume that for human interaction the input specified can only be of an approximate nature. Fuzzy systems use the notion of a membership function that distinguishes between the clear cases in a binary fashion, but models the uncertainties for some objects by a decreasing degree of membership. Though linguistic variables have often been employed to extract logical rules from data or assigning attributes to objects by evaluation of certain low level features (see e.g. [8]), their application on classical database retrieval has not yet been investigated. Also the area of operations research has since long studied the problem of letting users specify adequate weightings to express their preferences. In the context of decision making (see e.g. [12], [13], [16]) the notion of utility has been thoroughly investigated. Utility theory relates the desirability (in our case the scoring of an object) to the context (in our case the relation of dimensions in the combining function). Based on von Neumann – Morgenstern theory complex models have been proposed to assess utilities to each alternative, getting a proposal for a decision with maximized utility. However, the choice of the 'right' model is always left to empiric studies.

6. Summary and Outlook

We have presented several new results gained from the evaluation of a prototype of a personalized route planning system. Based on a preference framework as strict partial orders we showed how to integrate qualitative and quantitative preference constructors due to situational aspects and personal preferences. Using linguistic variables the mobile user can express his or her personalized queries, which are automatically mapped onto top-k queries with a weighted sum combining function. We analyzed the runtime efficiency of the SR-Combine algorithm to evaluate such top-k queries. Supplying SR-Combine with the exact knowledge about its cost-ratio of random to sorted accesses (being 1 : 16), it automatically tunes its performance within the given system architecture. Moreover, we analyzed the stability of the query results by several test series, systematically varying the mappings of the linguistic variables. It turned out that the stability of the result set could be improved by choosing mappings that provide increasingly higher weightings for the user-provided information about the importance of each characteristic. We also presented a geometric model allowing to explain this behavior.

However, such analytic stability measures can only provide a part of the picture. Following our investigation users are enabled to rely on reasonable system-provided mappings for linguistic variables, instead of having to choose random weightings like 'A is 0.56-times better than B'. However, there is a difference between a system behaving reasonably well and a system really delivering all relevant answers. To improve relevance more research in the area of cognitive expansion of queries and situation dependent preference profiles has to be invested.

7. References

[1] R. Agrawal, E. Wimmers. A Framework for Expressing and Combining Preferences. *Intern. Conf. on Management of Data (SIGMOD'00)*, Dallas, USA, 2000.

[2] W.-T. Balke, U. Güntzer, and W. Kießling. On Real-time Top k Querying for Mobile Services, *Intern. Conf. on Cooperative Information Systems (CoopIS'02)*, Irvine, 2002.

[3] W-T. Balke, W. Kießling, C. Unbehend. A situation-aware mobile traffic information prototype. In *Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS-36)*, Big Island, Hawaii, USA, IEEE Computer Society Press, 2003.

[4] W-T. Balke, W. Kießling, C. Unbehend. Personalized Services for Mobile Route Planning: A Demonstration. In *Proceedings of the 19th International Conference on Data Engineering (ICDE 2003)*, Bangalore, India, IEEE Computer Society Press, 2003.

[5] K. Böhm, M. Mlivoncic, H-J. Schek, R. Weber: Fast Evaluation Techniques for Complex Similarity Queries. *Intern. Conf. on Very Large Databases*, Rome, Italy, 2001.

[6] N. Bruno, L. Gravano, A. Marian. Evaluating Top-k Queries over Web-Accessible Databases. *Intern. Conf. on Data Engineering (ICDE'02)*, San Jose, CA, USA, 2002.

[7] J. Chomicki. Querying with intrinsic preferences. *Intern. Conf. on Advances in Database Technology (EDBT)*, Prague, Czech Republic, 2002.

[8] W. Duch, R. Adamczak, K. Grabczewski. Neural optimization of linguistic variables and membership functions. In *Proc. of the Int. Conf. on Neural Information Processing (ICONIP'99)*, Perth, Australia, 1999.

[9] R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. *Symposium on Principles of Database Systems(PODS'01)*, Santa Barbara, USA, 2001.

[10] R. Goldman, J. Widom. WSQ/DSQ, a practical approach for combined querying of databases and the web. *Intern. Conf. on Management of Data (SIGMOD'00)*, Dallas, USA, 2000.

[11] U. Güntzer, W-T. Balke, and W. Kießling. Optimizing multi-feature queries for image databases. *Intern. Conf. on Very Large Databases (VLDB'00)*, Cairo, Egypt, 2000.

[12] V. Hristidis, N. Koudas, Y. Papakonstantinou. PREFER: A System for the Efficient Execution of Multi-parametric Ranked Queries. *Intern. Conf. on Management of Data (SIGMOD'01)*, Santa Barbara, USA, 2001.

[13] R. Keeney, H. Raiffa. Decisions with Multiple Objectives: Preferences and Value Tradeoffs. Wiley, 1976.

[14] W. Kießling. Foundations of Preferences in Database Systems. *Intern. Conf. on Very Large Databases (VLDB'02)*, Hong Kong, China, 2002.

[15] W. Kießling, G. Köstler. Preference SQL - Design, Implementation, Experiences, *Intern. Conf. on Very Large Data Bases*, Hong Kong, China, 2002.

[16] J. Kozielecki. Psychological Decision Theory. Kluwer Academic Publishers, 1981.

[17] A. Leubner, W. Kießling. Personalized Keyword Search with Partial-Order Preferences, *Brazilian Symp. on Database Systems (SBBD)*, Gramado, Brazil, 2002.

[18] M. McGeachie, J. Doyle. Efficient Utility Functions for Ceteris Paribus Preferences. In *Proc. of Conf. on Artificial Intelligence and Conf. on Innovative Applications of Artificial Intelligence (AAAI/IAAI'02)*, Edmonton, Canada, 2002.

[19] M. Wagner, W-T. Balke, and W. Kießling. An XML-based Multimedia Middleware for Mobile Online Auctions. In J.Filipe, Sharp, B. and Miranda, P. (Eds.) *Enterprise Information Systems III*, Kluwer Academic Publishers, The Netherlands, 2002.

[20] L. Zadeh. The Concept of A Linguistic Variable and Its Application to Approximate Reasoning, *Information Sciences*, Vol. 8, 1975.