Developing Preference Band Model to Manage Collective Preferences

Wilfred Ng

Department of Computer Science and Engineering
The Hong Kong University of Science and Technology
Hong Kong
wilfred@cse.ust.hk

Abstract. Discovering user preference is an important task in various database applications, such as searching product information and rating goods and services. Previous work addressing user preference in databases either assumes that users are able to formulate explicit preference criteria when querying, which may not be feasible in reality, or develops mining techniques to discover users' implicit preference from their track record of purchasing and then to generate a restrictive linear ranking on items, which needs to process a huge amount of data possibly with noise. However, there still lacks of a unifying model that is able to capture both implicit and explicit user preference information and to support managing, querying and analysing the information obtained from different sources.
In this paper, we present a framework based on our newly proposed Preference Band Model (PBM), which aims to achieve several goals. First, the PBM can serve as a formal basis to unify both implicit and explicit user preferences. We develop the model using a matrix-theoretic approach. Second, the model provides means to manipulate different sources of preference information. We establish a set of algebraic operators on Preference-Order Matrices (POMs). Third, the model supports direct querying of collective user preference and the discovery of a preference band. Roughly, a preference band is a ranking on sets of equally preferred items discovered from a POM to classify collective user preference. We demonstrate the applicability of our framework by studying two real datasets.

1 Introduction

Discovering user preference is an important task in various database applications, such as searching product information and rating goods and services [3, 9, 11, 11]. Many business activities also rely heavily on estimating overall user preference in order to import user preferred goods and to design appropriate selling tactics. Previous work that incorporates user preference into databases mainly falls into two main categories of approaches. It either assumes that users are able to formulate their preference explicitly in terms of formulas or constraints [3, 10, 11], which may not be easy in reality, or develops mining techniques to discover implicit users' preference from the log of item selection [12] and then generate an adaptive full ranking of items, which may be too restrictive. However, there still lacks of a unifying model that is able to capture both implicit and explicit user preference information and to support managing, querying and analysing the information obtained from different sources.

We identify three problems that arise from handling preferences obtained from different sources and times. First, individual user preference can be modelled as a partial or full ranking between items [10, 11]; but how do we model a large amount of such ranking information, which represents very different, possibly noisy and conflicting user preferences? Second, how do we compare and contrast expected preference and real user preference? This also gives challenges to focus on some target preference data for analysing. Third, how do we rank and classify collective user preference and express the result in some form of simple but useful knowledge?

In this paper, we present a holistic framework shown in Figure 1, which involves a formal model, a set of algebraic operators and various algorithmic techniques to tackle the above problems. The framework aims to manage preference information in a systematic way and to support better analyses of collective user preferences.

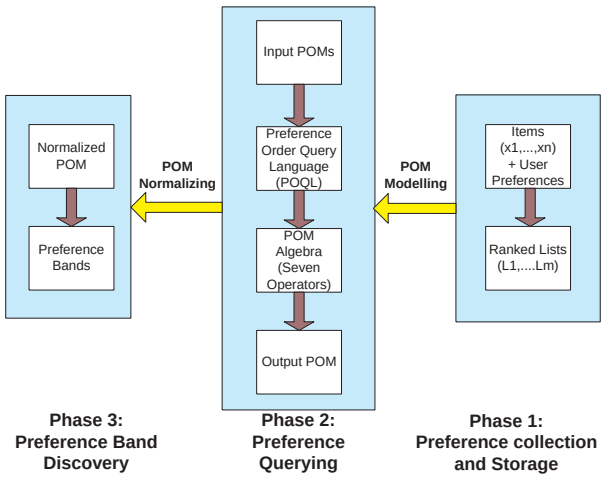

Fig. 1. The framework for managing collective user preference based on PBM

The main idea of our approach is first to convert a collection of user ranked items into a Preference-Order Matrix (POM). An entry a_{ij} in a POM is the frequency that one item x_i (more preferred) precedes another item x_j (less preferred) in the collected information, which captures the fact that in how many choices users preferred x_i to x_j. We also propose to use a declarative language termed Preference-Order Query Language (POQL) to formulate queries on POMs. A POQL expression can be transformed into a set of POM operators that support combining preference information in various ways. Then, we discover new preference knowledge in terms of the best possible preference band over a normalized POM. In a nutshell, a preference band is a linear order of sets of equally preferred elements found in the collective preference information.

As shown in the blueprint presented in Figure 1, we classify the process based on our Preference Band Model (PBM) into three phases. In the first phase, the PBM integrates implicit and explicit preference information represented by a set of user ranked lists, each of which is a user preference with possible choice order on target items. The

information is modelled within a unifying matrix structure called POMs. In the second phase, we use a high level declarative query language called POQL to support manipulation of preference data and to generate a more effective POM for further analyses. The language is executed and can be optimized via a set of POM operators, which shares the similar spirit of execution of SQL expressions in relational databases. In the third phase, we need to normalize the POM in order to have it run in an adapted PIVOT algorithm [1], which classify and rank items into a preference band. The algorithm is known to be an efficient and effective method for discovering order knowledge from similar matrix structures, which is capable of tolerating noise of order in reasonable limit [7, 6].

Our main contribution in this paper is the development work of the PBM that supports managing user preference information and discovering preference bands.

- We formalise the novel concept of preference-order matrices (POMs) that express collected preference information. POMs support the discovery of preference bands that represent the maximal consistent collective preference obtained from a given set of user preference ranked lists.
- We establish a new set of seven algebraic operators such as the sum, the union, the intersection and the selection on POMs. The intuitions of overall, change, and common preferences can be formalized by using them.
- We develop a new declarative language POQL based on the POM operators. The POQL can be transformed into a corresponding sequence of POM operations, which paves the way to study further optimization of running a POQL expression.
- We identify efficient algorithmic techniques for discovering preference bands, which shows that the discovery of preference bands from a POM is feasible in practice.
- Finally, we present the preliminary results of two interesting applications, which show the idea of preference bands is easily applicable.

The rest of the paper is organised as follows. In Section 2, we present some preliminary concepts of user preference. In Section 3, we present a set of formal POM operators. In Section 4, we discuss the issues arising from the three phases of the PBM framework. In Section 5, we present our results of finding preference bands on two applications. In Section 6, we review some related work on user addressing preference and the algorithmic techniques for finding preference bands. In Section 7, we give our concluding remarks.

2 Preliminaries

In this section, we first introduce the basic concepts of preference-order matrices (POMs) and preference bands used in the underlying model, and then define the problem of preference band discovery with respect to collective user preference.

User preferences can be expressed in an implicit and explicit order of items. For example, explicit user preference is formulated in a lexicographic order of preference attributes [11]. Preference SQL [9] is equipped with a "preferring" clause that allows user to specify soft constraints reflecting multiple preference terms. All these approaches would result in a preference ranking of items. Implicit preferences are some ranking order that is inferred from users' track record of choosing items against some background of item order [12], which also depends on the interpretation of the chosen items.

We assume that a preference rank list can be obtained via some users' implicit or explicit preference information and formalize the concept as follows:.

Definition 1. (Preference Ranked List) Given a set of items $I = \{x_1, \ldots x_n\}$ with an imposed order $<_I$. A choice $C \subseteq I$ is a list of items ordered according to some user preference $<_C$, where $(C, <_C)$ may or may not equal to $(I, <_I)$. We generate a *preference ranked list* T by appending $(I - C)$ into C such that $T = \langle C, (I - C)\rangle$, in which the items of $(I - C)$ are ordered according to $<_I$. We denote by $L = \{T_1, \ldots, T_m\}$ a collection of m preference ranked lists.

Note that in Definition 1, $<_C$ can represent the order of choosing items in C by the user, which is an implicit preference, or it can represent the order according to some explicit preference criteria. L is a collective user preference on I. The following example helps illustrate the ideas of individual and collective preferences.

Example 1. Let $I = \{a, b, c, d, e, f\}$ be the items stored in a database and there are two customers search some items on the web. Let the items be ranked according to the default order $I = \{a > b > c > d > e > f\}$, which can be interpreted as an arbitrary ranking or a ranking derived from some commercial considerations. One customer explicitly states his or her preference criteria, which result in a preferred ranking $C_1 = \{c > f > e > d > b > a\}$. Another customer does not state any preference and just browses the items ranked by the default order $<_I$. However, s/he only checks a subset of items of I in some different order given by $C_2 = \{d > b > f > e\}$.

We now show in Figure 2 two sets of preference ranked lists L_1 and L_2, each of which contains five preference ranked lists. L_1 and L_2 can represent collective customer preference information obtained from two sources. It can be checked that T_1 and T_8 are the preference ranked lists derived from C_1 and C_2 respectively. T_5 can also be viewed as a precise match between a user's preference and the shop's estimated user preference that is expressed in the default item order.

$$L_1 = \begin{array}{|l|}\hline T_1\text{: } c > f > e > d > b > a \\ \hline T_2\text{: } a > c > b > d > f > e \\ \hline T_3\text{: } a > c > d > b > f > e \\ \hline T_4\text{: } d > f > c > a > b > e \\ \hline T_5\text{: } a > b > c > d > e > f \\ \hline \end{array} \quad L_2 = \begin{array}{|l|}\hline T_6\text{: } c > f > e > d > b > a \\ \hline T_7\text{: } a > c > b > d > f > e \\ \hline T_8\text{: } d > b > f > e > a > c \\ \hline T_9\text{: } d > f > c > a > e > b \\ \hline T_{10}\text{: } b > d > f > e > c > a \\ \hline \end{array}$$

Fig. 2. An example of two sets of five preference ranked lists

A collection of preference ranked lists L from a given source can be modelled as a Preference Order matrices (POMs) M_L, which is a matrix structure derived from L to support further querying and preference band discovery.

Definition 2. (Preference Order Matrix) Given L and let $\mid L \mid = n$. A *Preference-Order Matrix* (POM) M_L with respect to L (or simply M if L is understood) is an $n \times n$ matrix such that each entry $a_{ij} \in M$ is equal to the number of occurrences that item x_i precedes item x_j in L. We define $a_{ij} \in M$ to be $\mid L \mid$ whenever $i = j$. M is said to be *normalized* if each entry $norm(a_{ij}) = \frac{a_{ij}}{n}$ (or equivalently, $\frac{a_{ij}}{a_{ii}}$ or $\frac{a_{ij}}{a_{jj}}$) whenever $i \neq j$, and $norm(a_{ij}) = \frac{1}{2}$ whenever $i = j$.

Clearly, it follows from Definition 2 that $0 \leq a_{ij} \leq \mid L \mid, a_{ij} + a_{ji} = \mid L \mid$ for any distinct i, j. The normalization implies that each entry $a_{ij} \in norm(M_L)$ is equal to the probability that item x_i precedes item x_j in L. Given $norm(M_L)$. The entries satisfies the following conditions: (1) (normalization constraint) $0 \leq a_{ij} \leq 1$; (2) (linearity constraint) $a_{ij} + a_{ji} = 1$; and (3) (triangle constraint) $a_{ij} \leq a_{ik} + a_{kj}$.

Normalised POMs are important. Although for presentation simplicity we assume all T being a total order, if a partial order is resulted from more general user preferences [10] we still can use a corresponding set of linear extensions to represent the order whose information of equally preferred items, say x_i and x_j, can still be captured by normalised POMs in the entry $a_{ij} = \frac{1}{2}$. Specifically, a partial order can be directly represented by a normalised POM as follows: $a_{ij} \in M$ is equal to 1 if $x_i \leq x_j$, $a_{ij} \in M$ is equal to 0 if $x_j \leq x_i$, and $a_{ij} \in M$ is equal to $\frac{1}{2}$ otherwise (i.e. x_i and x_j are incomparable). In other words, normalised POMs are flexible enough to capture more general user preference information.

Example 2. We now continue Example 1 to derive the POMs and their normalized form from L_1 and L_2. The POMs M_1 and M_2 and their normlized forms are given in Figures 3(a) and (b).

	a	b	c	d	e	f
a	5	4	3	3	4	3
b	1	5	1	2	4	3
c	2	4	5	4	5	4
d	2	3	1	5	4	4
e	1	1	0	1	5	1
f	2	2	1	1	4	5

$\Rightarrow$

	a	b	c	d	e	f
a	0.5	0.8	0.6	0.6	0.8	0.6
b	0.2	0.5	0.2	0.4	0.8	0.6
c	0.4	0.8	0.5	0.8	1.0	0.8
d	0.4	0.6	0.2	0.5	0.8	0.8
e	0.2	0.2	0.0	0.2	0.5	0.2
f	0.4	0.4	0.2	0.2	0.8	0.5

(a) M_1 and $norm(M_1)$

	a	b	c	d	e	f
a	5	2	2	1	2	1
b	3	5	2	2	3	3
c	3	3	5	2	3	2
d	4	3	3	5	4	4
e	3	2	2	1	5	0
f	4	2	3	1	5	5

$\Rightarrow$

	a	b	c	d	e	f
a	0.5	0.4	0.4	0.2	0.4	0.2
b	0.6	0.5	0.4	0.4	0.6	0.6
c	0.6	0.6	0.5	0.4	0.6	0.4
d	0.8	0.6	0.6	0.5	0.6	0.8
e	0.6	0.4	0.4	0.2	0.5	0.0
f	0.8	0.4	0.6	0.2	1.0	0.5

(b) M_2 and $norm(M_2)$

Fig. 3. The POMs M_1 and M_2 obtained from L_1 and L_2, and their normalized counterparts

One objective of this work is to establish a set of operators that are able to manipulate POMs and support in-depth analyses of collective user preference modelled in POMs. Another objective is to discover preference bands obtained from a given normalised POM. A preference band is intuitively a linear order of sets of equally preferred items taking into collective user preference. The motivation of defining a preference band is that in many adaptive searching applications, modelling the preference with a total order might be too restrictive in the sense that all items in I should be comparable. However, it is also too complex to manage, analyse and present the preference information expressed in general partial order.

We now give the formal definition of a preference band as follows.

Definition 3. (Preference Band) A *preference band* B of I with granularity k is an ordered partition of I given by $\langle P_1, \ldots, P_k \rangle$, where $P_i \subseteq I$ is a non-empty set called a band element in B. A band order of the items in I arising from B, denoted as $<_B$, can be inferred from B as follows: For any pair of items x and y in the same band element P, x and y are incomparable (i.e. $x \not<_B y$ and $y \not<_B x$). However, if x and y are from two distinct band elements P_i and P_j such that $i < j$, then $x <_B y$.

Essentially, a preference band adopts an appropriately low granularity to eliminate noises and ranking contradictions of items among preference ranked lists collected in L. However, there is an inherent cost of losing some information of the preference orders in individual T when putting items in a band element; as an extreme we can put all items in one single band element which is obviously no use. Thus, we need to establish a penalty cost function for measuring the quality of a preference band. A simple one is adopted as below but more sophisticated functions can be studied as a future work.

If M_1 and M_2 are two normalized POMs, we define the distance measure given by $D(M_1, M_2) = \Sigma_{i \neq j} \mid a_{ij}^{M_1} - a_{ij}^{M_2} \mid$.

Given a normalised POM M. We aim to find the best preference band describing M. We now formulate the Preference Band Discovery (PBD) problem.

Definition 4. (PBD problem) Let M_L be a normalised POM on I obtained from L and M_B be a normalised POM obtained from a preference band B of I. Find a preference band such that $D(M_L, M_B)$ is minimized. (Note that B is a partial order and can be represented by POM M_B.)

In other words, the main task of PBD is to find a preference band B on the items of I such that the penalty cost function is minimized, which is equal to the sum of values in the following three cases: (i) $\mid (a_{ij})_M - 1 \mid$ if $x_i <_B x_j$, (ii) $\mid (a_{ij})_M \mid$ if $x_j <_B x_i$, and (iii) $\mid (a_{ij})_M - \frac{1}{2} \mid$ otherwise.

Example 3. Consider our running example L_1 and L_2 given in Figure 2. Suppose there is a preference band $B = \langle \{a, b, c, d\} > \{e, f\} \rangle$ (which may not be the best one according to D). Then, it can be checked that $D(M_{L_1}, M_B) = 6.6$ and $D(M_{L_2}, M_B) = 10$. The difference is reasonable, since it can be checked that L_1 has a noise level of 40% with respect to the preference band B because two of the five preference ranked lists, namely T_1 and T_4, are not the linear extension [7] of B, whereas L_2 has an even worse noise level of 80% with respect to B because only T_7 is the linear extension of B. This motivates us to develop an effective algorithm to generate B from a given M_L.

3 The POM Operators

In this section, we define seven POM operators, namely the sum, the union, the difference, the intersection, the complement, the projection and the selection, each of which takes one or two given POMs as input parameters, and returns a POM. The operators serve as a useful tool to manipulate preference ranked lists and provide a basis to develop a high level query language. For example, the output result provides a deeper insight for sellers to check if the estimated preference order achieves the customer preference. The seven operators are briefly described as the table given in Figure 4.

The POM operators are easy to understand and to compute and their output result serves as a basis for carrying out the discovery of preference band. The sum operator is additive to preference data, that is, the result is incremental with respect to preference information. This gives advantage to handle preference data in a progressive manner, say temporal change in preference can be detected. The overall, change, common, opposite preference can be formalised by the sum, the union, the difference, the intersection,

and the complement operations. We can focus on the preference information of a particular set of items by using the projection and the selection operations. On the other hand, matrices are an elegant notion studied in a well-established branch of mathematics. It implies that many interesting results in matrix theory can be used to strengthen the foundation of PBMs as a development work.

Operators	Functions
Sum (+)	To add up the preference orders that are inferred from two preference ranked lists.
Union ($\cup$)	To overlap the preference orders that are inferred from two preference ranked lists.
Difference ($-$)	To differ the preference orders that are inferred from two preference ranked lists.
Intersection ($\cap$)	To obtain the common preference orders that are inferred from two preference ranked lists.
Complement ($\neg$)	To obtain the preference orders that are opposite to that are inferred from the preference ranked list.
Projection (π_X)	To extract the preference orders that are only related to the items $X \subseteq I$ and are inferred from the preference ranked list.
Selection (σ_p)	To select the preference orders that satisfy the predicate p and are inferred from the preference ranked list.

Fig. 4. Brief description of the seven POM operators

We need two binary operators, denoted as *min* and *max*, to represent the usual minimum and maximum of two given integers in defining the POM operators. From now on, we use throughout the paper M_1 and M_2 to represent two POMs defined over the same set of items I. The sum operation is given in Definition 5.

Definition 5. (Sum) The *sum* of two POMs M_1 and M_2 , denoted as $M_1 + M_2$, is defined as a POM M_3 over I such that for all $i, j \in \{1, \ldots, n\}, (a_{ij})_3 = (a_{ij})_1 + (a_{ij})_2$.

An interesting property of the sum operation is that the sum of two POMs M_1 and M_2, which originates from the two respective preference ranked lists L_1 and L_2, is equal to the matrix which originates from the preference ranked list L_3 containing the combination of the information in L_1 and L_2 (i.e. $L_3 = L_1 \cup L_2$). This implies that we are able to perform analyses on "overall" preference information by summing up "individual" pieces of preference information, in this sense we say that the sum operator is *additive* with respect to the preference data.

Another possibility to combine preference information is to consider the maximum number of occurrences of precedence between items, taking into account the "overlap" effect. We now define this operation by the union operator as follows.

Definition 6. (Union) The *union* of two POMs M_1 and M_2, denoted as $M_1 \cup M_2$, is defined as a POM M_3 over I such that $(a_{ii})_3 := max((a_{ii})_1, (a_{ii})_2)$ for all $i \in \{1, \ldots, n\}$ and for all pairs of distinct $i, j \in \{1, \ldots, n\}$, we obtain $(a_{ii})_3$ as follows:
Let $x := max((a_{ij})_1, (a_{ij})_2)$ and $y := max((a_{ji})_1, (a_{ji})_2)$.
If $x > y$ then $(a_{ij})_3 := x$ and $(a_{ji})_3 := max((a_{ii})_1, (a_{ii})_2) - x$;
otherwise (i.e. $x \leq y$) $(a_{ji})_3 := y$ and $(a_{ij})_3 := max((a_{ii})_1, (a_{ii})_2) - y$.

Note that the comparison $x > y$ makes sure that the most dominant precedence in M_1 and M_2 is chosen as the output. In contrast, we can consider the less dominant precedence but it is common to both M_1 and M_2. This becomes another operation called intersection defined as follows.

Definition 7. (Intersection) The *intersection* of two POMs M_1 and M_2, denoted as $M_1 \cap M_2$, is defined as a POM M_3 over I such that $(a_{ii})_3 := min((a_{ii})_1, (a_{ii})_2)$ for all $i \in \{1, \ldots, n\}$ and for all pairs of distinct $i, j \in \{1, \ldots, n\}$, we obtain $(a_{ii})_3$ as follows:

Let $x := max((a_{ij})_1, (a_{ij})_2)$ and $y := max((a_{ji})_1, (a_{ji})_2)$.
If $x < y$ then $(a_{ij})_3 := x$ and $(a_{ji})_3 := min((a_{ii})_1, (a_{ii})_2) - x$;
otherwise (i.e. $x \geq y$) $(a_{ji})_3 := y$ and $(a_{ij})_3 := min((a_{ii})_1, (a_{ii})_2) - y$.

We now introduce the difference operator, which is useful to contrast two preference ranked lists. For example, we can use the difference operator to find the temporal change in preference data obtained at two different time intervals or to compare the preference data obtained from two user groups having different preference profiles.

Definition 8. (Difference) Assume $\mid L_1 \mid > \mid L_2 \mid$. The *difference* of two POMs M_1 and M_2 , denoted as $M_1 - M_2$, is defined as a POM M_3 over I such that $(a_{ii})_3 := (a_{ii})_1 - (a_{ii})_2$ for all $i \in \{1, \ldots, n\}$ and for all pairs of distinct $i, j \in \{1, \ldots, n\}$, we obtain $(a_{ii})_3$ as follows:

Let $x := (a_{ij})_1 - (a_{ij})_2$ and $y := (a_{ji})_1 - (a_{ji})_2$.
Case $x > 0$ and $y > 0$: $(a_{ij})_3 := x$ and $(a_{ji})_3 := y$;
Case $x > 0$ and $y \leq 0$: $(a_{ij})_3 := min(x, (a_{ii})_1 - (a_{ii})_2)$ and $(a_{ji})_3 := (a_{ii})_1 - (a_{ii})_2 - min(x, (a_{ii})_1 - (a_{ii})_2)$;
Case $y > 0$ and $x \leq 0$: $(a_{ij})_3 := (a_{ii})_1 - (a_{ii})_2 - min(y, (a_{ii})_1 - (a_{ii})_2)$ and $(a_{ji})_3 := min(y, (a_{ii})_1 - (a_{ii})_2)$.

Notably, the operator is undefined for $\mid L_1 \mid \leq \mid L_2 \mid$. This prevents the happening of meaningless negative entries in a POM. Thus, there is no need to consider the possibility of $y \leq 0$ and $x \leq 0$ in the definition. It also implies that either x or y is strictly positive as shown in the three cases in Definition 8.

The following is an interesting operator that defines the "reverse" of the preference order for all items in I.

Definition 9. (Complement) The *complement* of a POM M_1, denoted as $\neg M_1$, is defined as a POM M_2 over I such that for all $i, j \in \{1, \ldots, n\}$, $(a_{ij})_2 = (a_{ji})_1$.

Similar to using relational algebra [2], we may focus on a target set of items for further analyses. The following two operators aim to realise the similar objective. Unlike operating on relational tables, we still need to preserve the entries of the non-target items in the output, since POM is a matrix. To tackle this problem, we simply fill in the entries of these items according to the default order $<_I$. I.e. All the items in $(I - X)$ in the projection are ordered according to $<_I$ in the output POM.

Definition 10. (Projection) The *projection* of a POM M_1, denoted as $\pi_X(M_1)$ where $X \subseteq I$ is a set of items, is defined as a POM M_2 as follows:

Case $x_i, x_j \in P$: $(a_{ij})_2 = (a_{ij})_1$.

Case $x_i \in P$ but $x_j \notin P$: $(a_{ij})_2 = (a_{ij})_1 + (a_{ji})_2$ and $(a_{ji})_2 = 0$.

Case $x_i, x_j \notin P$: $(a_{ij})_2 = (a_{ij})_1 + (a_{ji})_2$ if $x_j <_I x_i$; otherwise (i.e. $x_i <_I x_j$) $(a_{ji})_2 = 0$.

The tricky point in defining the POM selection is that we should not allow arbitrary entry comparison, since it is difficult to know or seldom need to concern the absolute occurrence of precedence in L. Instead, we choose the relative ratio of precedence between items as the parameter used in the selection predicate, which always falls in a unit interval. This also captures the intuition that how much x is more preferred to y.

Definition 11. (Selection) The *selection* of a POM M_1, denoted as $\sigma_{\theta x}(M_1)$, where θ is a comparator such as $>$, $<$, $\geq$, $\leq$, or $=$, and $x \in [0, 1]$ is a positive number, is defined as a POM M_2 as follows:

For all $i \in \{1, \ldots, n\}(a_{ii})_2 = (a_{ii})_1$ and for all distinct $i, j \in \{1, \ldots, n\}$,

If $\mid \frac{(a_{ij})_1 - (a_{ji})_1}{(a_{ij})_1 + (a_{ji})_1} \mid \theta x$, then $(a_{ij})_2 = (a_{ij})_1$;

otherwise (i.e. $\mid \frac{(a_{ij})_1 - (a_{ji})_1}{(a_{ij})_1 + (a_{ji})_1} \mid \bar{\theta} x$) if $x_j <_I x_i$ $(a_{ij})_2 = (a_{ij})_1 + (a_{ji})_2$; or else (i.e. $x_i <_I x_j$) $(a_{ji})_2 = 0$,

where the notation $\bar{\theta}$ denotes the complement of θ (e.g. $\bar{\theta}$ is "$\leq$" when θ is "$>$").

It follows from Definition 11 $\sigma_{\geq 0}(M) = M$, where all entries are trivially selected. The result $\sigma_{>1}(M)$ represents the preference being reduced to the default order $<_I$. This can be used as the *identity* POM ID for revealing some interesting properties of the union and intersection operators in the following discussion.

We let $\delta_1 \in \{+, \cup, \cap\}$ and $\delta_2 \in \{\neg, \pi_X\}$ and $\delta_3 \in \{\neg, \sigma_p, \pi_X\}$. The following properties of the POM operations can be verified by Definitions 5 to 11.

Associative Property. The POMs on the same I are associative under *sum*, *union* and *intersection* which is shown in the equations as follows:

$(M_1 \delta_1 M_2)\delta_1 M_3 = M_1 \delta_1 (M_2 \delta_1 M_3)$.

Commutative Property. The POMs on the same I are also commutative under *sum*, *union* and *intersection*.

(1) $M_1 \delta_1 M_2 = M_2 \delta_1 M_1$. (2) $\delta_3'(\delta_3(M)) = \delta_3(\delta_3'(M))$.

Distributive Property. *Projection* and *negation* are distributive under *sum*, *union* and *intersection*.

$\delta_2(M_1\ \delta_1\ M_2) = (\delta_2(M_1))\ \delta_1\ (\delta_2(M_2))$.

In general, $\sigma_p(M_1 \delta_1 M_2) \neq (\sigma_p(M_1))\delta_1(\sigma_p(M_2))$ and $\pi_X(M_1 - M_2) \neq (\pi_X(M_1)) - (\pi_X(M_2))$. However, $\neg(M_1 - M_2) = (\neg(M_1)) - (\neg(M_2))$ and $\neg(\neg(M)) = M$.

We now let $\sigma_{>1}(M) = ID$ and present the identity property.

Identity Property. *Union* and *intersection* satisfy the identity property.

(1) $M \cup ID = ID$. (2) $M \cap ID = M$.

Example 4. We now make use the POMs M_1 and M_2 given in Figure 3 to show some of the results of $M_1 \delta_1 M_2$ and $\delta_3(M_2)$ in Figure 5.

4 Implementation Issues

In this section, we discuss the techniques and methods that addresses the challenges arising from the three phases of the PBM framework depicted in Figure 1.

	a	b	c	d	e	f
a	10	6	5	4	6	4
b	4	10	3	4	7	6
c	5	7	10	6	8	6
d	6	6	4	10	8	8
e	4	3	2	2	10	1
f	6	4	4	2	9	10

(a)

	a	b	c	d	e	f
a	5	4	3	1	4	1
b	1	5	1	2	4	3
c	2	4	5	4	5	4
d	4	3	1	5	4	4
e	1	1	0	1	5	0
f	4	2	1	1	5	5

(b)

	a	b	c	d	e	f
a	5	2	2	3	2	3
b	3	5	2	2	3	3
c	3	3	5	2	3	2
d	2	3	3	5	4	4
e	3	2	2	1	5	1
f	2	2	3	1	4	5

(c)

	a	b	c	d	e	f
a	5	4	3	5	5	5
b	1	5	1	5	5	5
c	2	4	5	5	5	5
d	0	0	0	5	5	5
e	0	0	0	0	5	5
f	0	0	0	0	0	5

(d)

	a	b	c	d	e	f
a	5	4	5	5	4	5
b	1	5	1	5	4	5
c	0	4	5	4	5	4
d	1	0	1	5	4	4
e	1	1	0	1	5	1
f	0	0	1	1	4	5

(e)

Fig. 5. The results of the some binary and unary POM operations (a) $M_1 + M_2$ (b) $M_1 \cup M_2$ (c) $M_1 \cap M_2$ (d) $\pi_{abc}(M_1)$ (e) $\sigma_{\theta>0.2}(M_1)$

4.1 Phase 1: How to obtain and store POMs?

We can construct the POM M by using L in two ways. The first way is to adopt a brute force approach to bookkeep a_{ij} when scanning the total order in which x_i precedes x_j. As the algorithm for discovering preference bands needs to take a normalized POM as the input, we still need to compute the normalized values of the entries in M in the final phase. So we may use a more direct way to obtain $norm(M)$. We can sample possible orderings of the user preference $T \in I$ and in this case $norm(a_{ij})$ is set to be the fraction of L in which x_i precedes x_j, such as Markov Chain Monte Carlo method used in [15]. In this approach, we avoid the heavy computation to track all precedences. In fact it is inevitable to have noise in preference data in real applications. Sampling techniques seem to be more appropriate to generate normalized POMs.

The storage scheme of a matrix greatly affects the performance of POM operators. In practice, there are usually many items to be considered. As I is large and the number of preferred items is small, the POMs may be very sparse with respect to the entries that actually represent user preference data. As the entries of diagonal and lower triangular portion of a POM can be deduced from $\mid L \mid$, we may develop a concise version that targets on only those entries that contain preference information. Specifically, we fill in zeros in entry of (i) the lower triangular portion of the POM and (ii) the diagonal running top left to bottom right. The first condition is valid, since we have $a_{ji} =\mid L \mid -a_{ij}$. The second condition is also valid since we have $a_{ii} =\mid L \mid$. There will be more zero entries in the upper triangular portion of M if the ranking order is skew.

Example 5. We use M_1 given in Figure 3 to illustrate some storage schemes. The matrix is reduced into a concise version as shown in Figure 6. The first three rows of Figure 7 constitute a simple storage scheme for M_1, which is called the co-ordinate Based scheme (COO). It stores each nonzero entry together with its column and row indexes in three arrays as shown. The scheme holds the non-zero entries in row-first order.

In literature, the technique of storing sparse matrices has been intensively studied [13, 8]. There are other storage schemes which make use the zero entries to compress a matrix further such as the Compressed Sparse Row (CSR) storage scheme, which differs from COO in the Compressed Row array. In the table shown in Figure 7, the fourth row, namely the *compressed row* (54221), represents five "1" (meaning the first row), four "2" (meaning the second row), two "3" (meaning the third row), and so on. We fill in "-1" if there is no non-zero entry at a row if it is not the end (no occurrence

in this example). Another is the Compressed Sparse Column (CSC) storage scheme, which is similar to CSR but uses compressed column array to hold the location of the first non-zero entry of that column. There are also other sparse matrix storage schemes, such as Compressed Diagonal Storage (CDS) and Jagged Diagonal Storage (JDS) [14], which need further study on their effectiveness on storing and compressing POMs.

	a	b	c	d	e	f
a	5	4	3	3	4	3
b	1	5	1	2	4	3
c	2	4	5	4	5	4
d	2	3	1	5	4	4
e	1	1	0	1	5	1
f	2	2	1	1	4	5

$\Longrightarrow$

	a	b	c	d	e	f
a	0	4	3	3	4	3
b	0	0	1	2	4	3
c	0	0	0	4	0	4
d	0	0	0	0	4	4
e	0	0	0	0	0	1
f	0	0	0	0	0	0

Fig. 6. POM in concise form by filling in zeros

Non-zero	4	3	3	4	3	1	2	4	3	4	4	4	4	1
Column	2	3	4	5	6	3	4	5	6	4	6	5	6	6
Row	1	1	1	1	1	2	2	2	2	3	3	4	4	5
Compressed Row	5	4	2	2	1									

Fig. 7. Simple Co-ordinate-Based (COO) and Compressed Spares Row (CSR) Schemes

4.2 Phase 2: How to use the POM operators to formulate queries?

We develop a declarative language on POMs in our framework and term the langauge the Preference Order Query Language (or simply the POQL). A POQL expression is executed via the seven POM operators defined in Section 3, which shares the same principle of translating a SQL expression into a sequence of relational algebra operations. We now define the POQL syntax in Backus Naur Form (BNF):

<query> :: = <selectClause><fromClause>[<conditionClause>]
<selectClause> :: = SELECT <itemList>
<queryList> :: = query [, query. . .]
<fromClause> :: = FROM <matrixIdentifier> | FROM <operator> <matrixList>
<conditionClause> :: = WHERE PREFERENCE RATIO <compOp> a number in [0,1]
<itemList> :: = itemIdentifier [, itemIdentifier. . .] | $*$
<matrixList> :: = matrixIdentifier [, matrixIdentifier. . .]
<operator> :: = SUM|UNION|DIFFERENCE|INTERSECT|COMPLEMENT
<compOp> :: = |<=|>=|<|=

There are three main clauses in a query expression: the *select*, *from*, and *condition*. Among them the *select* and the *from* clauses are compulsory, while the *condition* clause is optional. Similar to SQL, POQL is a simple declarative language but is expressive enough to formulate query on finding preference information stored as POMs.

We execute a POQL expression by translating it into a sequence of POM operations using Algorithm 1. Suppose $X \subseteq L$ is a set of items which appears in the select clause, $\mathcal{M}$ is a set of m POMs which appears in the from clause, where $\mathcal{M} = \{M_1^*, M_2^*, \ldots, M_m^*\}$ and M^* means either M or *COMPLEMENT* M, and $OPER \in \{\epsilon, SUM, UNION, DIFFERENCE, INTERSECT\}$. Note that the input POQL query expression is assumed to be syntactically valid. If $OPER = \epsilon$, then $m = 1$.

Now, we present a set of examples, which illustrates the usage of the POQL expressions and the translation into the corresponding sequence of POM operations. Let M_1, M_2 and M_3 be the POMs obtained from different sources.

(Q_1): We want to know how popular the items x_1 and x_2 in the three branches of a shop (sources M_1, M_2 and M_3).

Algorithm 1 Translate POQL Algorithm

Input: A POQL expression q

LET q = <selectClause><fromClause>[<conditionClause>]

<selectClause> := "SELECT $X \mid *$"

<fromClause> := "FROM $OPER\ \mathcal{M}$"

<conditionClause> := "WHERE PREFERENCE RATIO θx"

Procedure:

Step 1 : For the fromClause, **CASE** $OPER$ **OF**:

$\epsilon : TEMP :=$ "M_1"

$COMPLEMENT : TEMP :=$ "$\neg M_1$"

$SUM : TEMP :=$ "$M_1 + M_2 + \cdots + M_m$"

$UNION : TEMP :=$ "$M_1 \bigcup M_2 \bigcup \cdots \bigcup M_m$"

$DIFF : TEMP :=$ "$M_1 - M_2 - \cdots - M_m$"

$INTERSECT : TEMP :=$ "$M_1 \bigcap M_2 \bigcap \cdots \bigcap M_m$"

Step 2 : For the selectClause:

IF "$*$" **THEN** $TEMP := TEMP$

ELSE $TEMP :=$ "$\pi_X(TEMP)$"

Step 3 : **IF** there is a whereClause **THEN** $TEMP :=$ "$\sigma_{\theta x}(TEMP)$"

Output: $TEMP$ expression

POQL expression: SELECT x_1, x_2 FROM SUM M_1, M_2, M_3.

POM operation: $\pi_{\{x_1,x_2\}}(M_1 + M_2 + M_3)$.

(Q_2): We want to find out the essential difference of preferences between the two groups of users in M_1 and M_2. We consider only extreme cases of those items having preference ratio > 0.9.

POQL expression: SELECT * FROM DIFFERENCE M_1, M_2 WHERE PREFERENCE_RATIO > 0.9.

POM operation: $\sigma_{>0.9}(M_1 - M_2)$.

(Q_3): We want to get the specific information of a particular set of items $X = \{x_1, x_2, x_3\}$ with common preference in the three sources. We consider only those items having roughly equal preference (ratio < 0.1).

POQL expression: SELECT X FROM INTERSECT M_1, M_2, M_3 WHERE PREFERENCE_RATIO < 0.1.

POM operation: $\sigma_{<0.1}(\Pi_X(M_1 \bigcap M_2 \bigcap M_3))$.

Let us consider the query Q_3. The optimal plan is to first execute the POM operators that can minimize the number of non-zero elements in the matrix. For the sake of efficiency, the *projection* should be executed as early as possible. So a better POM execution plan of Q_5 can be obtained as follows:

(Q_4): $\sigma_{<0.1}(\pi_X(M_1) \bigcap \pi_X(M_2) \bigcap \pi_X(M_3))$.

4.3 Phase 3: How to generate preference bands?

The challenge of this final phase is to tackle the PBD problem in Definition 4. However, the problem is equivalent to the Bucket Order Discovery (BOD) problem, which aims to find a linear order of buckets from a given set of linear orders. The BOD problem has

been proved to be NP-hard [1]. Thus, we have to resort to heuristic algorithms. Given a set of full rankings T_1 and T_2 over I. Gionis et al. [7] proposed a heuristic algorithm called Bucket Pivot, which is adapted from Ailon's randomized algorithm Pivot [1] (or simply called the PIVOT algorithm).

Essentially, the PIVOT algorithm starts with a random selected element and then compares the elements with others and finally generates three classes of "left", "same" and "right" classes. To apply the techniques in our context, the PIVOT algorithm can be employed to exploit precedence probabilities stored in the input normalized POM M and then to discover the output preference band recursively, in which a preference band can be regarded as a bucket order B. The three classes of buckets correspond to the lower band (LB), current band (CB) and upper band (UB) elements.

Specifically, the adapted PIVOT algorithm runs in a quick-sort-like manner. In each recursion, a random pivot item x_i is first selected and the following three band elements, $\langle \text{UB} > \text{CB} > \text{LB} \rangle$, are also created for x_i. Other items are then compared with the pivot x_i. An item x_j will be put into CB if the precedence probability a_{ij} satisfies the following inequality: $0.5 - \beta \leq a_{ij} < 0.5 + \beta$, where $\beta \in [0, 0.5]$ is a parameter that describes the degree of precision of precedence probability relative to 0.5. The drawback of using PIVOT is that it is not easy to set a suitable β value and the choice of β affects the final number of band elements in B. Clearly if $\beta = 0$ we cannot generate any approximated band element to provide insight of the collective preference in the input POM. On the other hand, if $\beta = 0.5$ we have a very imprecise (and is likely to be large) band element that contains all x_j having $a_{ij} \neq 1$ (which has a high probability). Fortunately, according to [7], the default value $\beta = 0.25$ is shown to be able to make PIVOT achieve good approximation ratio with respect to the penalty cost function D.

We now present the adapted PIVOT algorithm for PBD problem in Algorithm 2, which adapts PIVOT in our contest. We simply assume $\beta = 0.25$ for running PBD.

Algorithm 2 The Preference Band Discovery Algorithm $PBD(X, M, \beta)$

Input: A normalized POM M of I, $X \subseteq I$ and $\beta = 0.25$ by default
Procedure:
Step 1 : **IF** $X = \emptyset$ **RETURN** $\emptyset$
Step 2 : Pick $x_i \in I$ randomly as a pivot **DO**
$UB \leftarrow \emptyset$
$CB \leftarrow \{x_i\}$
$LB \leftarrow \emptyset$
Step 3 : **FOR ALL** items $x_j \in (X - \{x_i\})$ **DO**
IF $0.5 + \beta \leq a_{ij}$ **THEN** $UB \leftarrow UB \cup \{x_j\}$
ELSE IF $0.5 - \beta \leq a_{ij} < 0.5 + \beta$ **THEN** $CB \leftarrow CB \cup \{x_j\}$
ELSE IF $a_{ij} < 0.5 - \beta$ **THEN** $LB \leftarrow LB \cup \{x_j\}$
Output: A preference band given by $\langle PBD(UB, M, \beta), CB, PBD(LB, M, \beta) \rangle$

5 Applications of Preference Bands

We now present an application of the PBM to discover the user preference in (1) a real rating movie dataset and (2) a real clickstream dataset. A clickstream is the evidence of user preference in choosing a web page to browse.

(1) Movie Preference. We consider the ranking of movie obtained from Eachmovie data[1]. The dataset consists of 72916 users, 1628 movies, and 2811983 movie votes. Each of the movie votes has a weight and a score, where the weight denotes whether or not the user actually saw the movie, and where the score denotes the user's movie rating. Using Algorithm 2, we ignore the weight and set it to 1 for all users. There are actually very different preference in ranking the movies. We target on the following set of ten movies to find the PB:

$I = \{$Strange Days (SD), Lawnmover Man 2 (LM), Flintstones (FS), Free Willy (FW), Godfather (GF), 3 Musketeers (MU), 101 Dalmatians (DM), Empire Strikes Back (ES), Barb Wire (BW), and Jungle Book (JB)$\}$.

Figure 8 shows the normalised POM for the rankings of the 10 movies. We obtain the best preference band of four band elements of movies as follows: $\langle\{GF, ES\} > \{SD, JB, \} > \{FS, FW, DM, MU\} > \{BW, LM\}\rangle$. The band elements are not so trivial to be obtained from the huge amount of data or even from the POM given in Figure 8, which is obtained after the processing efforts in Phases 1 and 2.

$$\begin{bmatrix} & ES & GF & JB & SD & MU & DM & FW & FS & LM & BW \\ ES & 0.5 & 0.51 & 0.70 & 0.66 & 0.78 & 0.86 & 0.86 & 0.88 & 0.94 & 0.98 \\ GF & 0.49 & 0.5 & 0.69 & 0.66 & 0.76 & 0.87 & 0.87 & 0.89 & 0.93 & 0.98 \\ JB & 0.30 & 0.31 & 0.5 & 0.55 & 0.66 & 0.76 & 0.79 & 0.80 & 0.91 & 0.94 \\ SD & 0.34 & 0.34 & 0.45 & 0.5 & 0.63 & 0.71 & 0.77 & 0.74 & 0.88 & 0.92 \\ MU & 0.22 & 0.24 & 0.34 & 0.37 & 0.5 & 0.57 & 0.69 & 0.64 & 0.79 & 0.87 \\ DM & 0.14 & 0.13 & 0.24 & 0.29 & 0.43 & 0.5 & 0.62 & 0.57 & 0.78 & 0.79 \\ FE & 0.14 & 0.13 & 0.21 & 0.23 & 0.31 & 0.38 & 0.5 & 0.54 & 0.78 & 0.79 \\ FS & 0.12 & 0.11 & 0.20 & 0.26 & 0.36 & 0.43 & 0.46 & 0.5 & 0.70 & 0.73 \\ LM & 0.06 & 0.07 & 0.09 & 0.12 & 0.21 & 0.22 & 0.22 & 0.30 & 0.5 & 0.46 \\ BW & 0.02 & 0.02 & 0.06 & 0.08 & 0.13 & 0.15 & 0.21 & 0.27 & 0.54 & 0.5 \end{bmatrix}$$

Fig. 8. The normalized POM of movie preference to discover preference bands

(2) Web Page Preference. We consider the user page visits of msnbc.com on a single day. The clickstream dataset is obtained from MSNBC data[2]. Each user has exactly one sequence of page visits, where each page visit is recorded as a URL category. There are 17 categories in total, and each category is encoded by a distinct integer from 1 to 17. In other words, $I = \{1, \ldots, 17\}$ with usual numerical order and L consists of a set of T sequences, each of which is a permutation of I.

First, we keep the first occurrence of a category in each sequence T and denote the resulting sequence as C. Usually, $C \subseteq I$ and thus $\mid C \mid$ is less than 17. We denote the (17 - $\mid C \mid$) categories as S and sort the categories in S according to the numerical order of their integer code. Then, we append S to C to form a permutation of I as a result, which is a T sequence. We select a subset of the sequences whose $\mid C \mid$ contains at least 14 categories in order to increase the effect of the user preference in L. In total, we identify 160 such T sequences.

Examples of URL categories are "frontpage" and "msn-sports". The page visit sequence is a total order if we do not consider duplicate page visits. The order of page

[1] `http://research.compaq.com/SRC/eachmovie`

[2] `http://kdd.ics.uci.edu/databases/msnbc/msnbc.html`

visits reflects a user's browsing habit. By clustering page-visit orders, we generate a user preferred categorization. For example, in Figure 9, we may use a preference band to indicate the common browsing habits of the set of users, who may usually start from reading some news, either about politics or about sports, and end up with a look at the weather conditions.

User 1:	frontpage → news → sports → weather
User 2:	news → politics → weather
⋮ ⋮	⋱ ⋮
User n:	frontpage → politics → weather

Fig. 9. Collective users' browsing habits to form a preference band

The best preference band discovered in MSNBC data is found as follows: $\langle\{frontpage, news\} > \{tech, local, on\text{-}air, misc, weather, msn\text{-}news, health, living, business, sports, summary, travel\} > \{opinion\} > \{msn\text{-}sports, bbs\}\rangle$. It shows that the item "news" is put in the first band element in addition to the "frontpage". The result that both "frontpage" and "news" are in the top band element is not so trivial but seems to match real-life experience: while users can first visit "frontpage" and then go from the "frontpage" to "news", users may have bookmarked "news" and visit "news" directly. In contrast, if we use a preference ranking instead of a preference band to represent such browsing preference, "frontpage" would most likely be put in the top of the preference rank, which wrongly assumes that users normally start with the "frontpage" and then go to the "new". Thus, using a preference band is a more natural and meaningful representation of collective preference.

6 Related Work

Preferences are receiving much attention in querying, since DBMSs need to provide better information services in advanced applications [9, 10]. In particular, preference SQL [10] is equipped with a "preferring" clause that allows user to specify soft constraints reflecting multiple preference terms. Implicit preference have been commonly studied in the area of searching such as using clickthrough data to mine user preference in web browsing [12]. In reality, implicit and explicit user preferences are important but there lacks of a formal basis to accommodate and manipulate both of them.

Our previous work [11] models single user preference as a hierarchy of its underlying data values and formalise the notion of Prioritized Preferences (PPs). We then consider multiple user preferences in ranking tuples in a relational table. We examine the impact of a given set of PPs on possible choices in ranking a database relation and develop a new notion of Choice Constraints (CCs) in a relation.

The PBD problem can be translated into the bucket order discovery (BOD) problem studied in recent years. The problem of obtaining a single bucket order from a collection of input total orders has been considered by Fagin et al. [4, 5] in their general framework of comparing and aggregating partial rankings. In our recent work in [6], we develop a new algorithm, called GAP, to tackle the BOD problem. The GAP algorithm consists of a two phase ranking aggregation and involves the use of a novel rank gap heuristic for segmenting multiple quantile orders.

7 Concluding Remarks

In this paper, we propose a new PBM framework that consists of three phases of work for managing a collection of user preference data that are modelled as POM matrices. We also establish a set of operations which serve as the formal tool to analyse and manipulate preference data. We further develop a declarative query language based on the POM operators and a new concept of preference band to discover and classify collective user preference. We discuss various technical issues related to the three phases.

There are indeed many interesting issues that deserve further study. In the modelling aspect, we need to clarify if POQL is expressive enough to obtain important preference information from given POMs. In the deployment aspect, we still need to study the efficiency of the POM operators and devise effective optimization strategy for the execution of the operations. The algorithm of finding preference bands is also related to the research work of bucket order in the data mining area. In the application aspect, we believe there should be many more interesting possibilities to apply our framework, in addition to the usual database and web applications discussed in Section 5.

References

1. N. Ailon, M. Charikar, and A. Newman. Aggregating inconsistent information: Ranking and clustering. In: *ACM STOC*., pages 684–693, (2005).
2. P. Atzeni and V. De Antonellis. *Relational Database Theory*. Benjamin/ Cummings Publishing Company, Inc., (1993).
3. J. Chomicki. Preference formulas in relational queries. *ACM Transaction Database System* **28**(4): pages 427-466 (2003).
4. R. Fagin, R. Kumar, and M. Mahdian. Comparing and aggregating with ties. In: *ACM PODS*, pages 47–58, (2004).
5. R. Fagin, R. Kumar, and D. Sivakumar. Efficient similarity search and classification via rank aggregation. In: *ACM SIGMOD*, pages 301–312, (2003).
6. J. Feng, Q. Fang, and W. Ng. Discovering Bucket Orders from Full Rankings. To appear: *ACM SIGMOD*, (2008).
7. A. Gionis, H. Mannila, K. Puolamaki, and A. Ukkonen. Algorithms for discovering bucket orders from data. In: *ACM SIGKDD*, pages 561–566, (2006).
8. N. Goharian, A. Jain, and Q. Sun. Comparative analysis of sparse matrix algorithms for information retrieval. *Journal of Sys., Cyb. and Inf.*, 1(1), (2003).
9. W. Kießling and G. Köstler. Preference SQL - Design, Implementation, Experiences. In: *Proc. of VLDB*, (2002).
10. W. Kießling and G. Köstler. Foundations of Preference in Database Systems. In: *Proc. of VLDB*, (2002).
11. W. Ng. *Prioritized Preferences and Choice Constraints*. In: Proc of ER 2007, LNCS Vol. 4801, pp. 261-276, (2007).
12. Q. Tan et al. *Applying Co-training to Clickthrough Data for Search Engine Adaptation.* In: Proc. of DASFAA, LNCS Vol 2973, pages 519-532, (2004).
13. N. Ahmed et al. A framework for sparse matrix code synthesis from high-level specifications. In: *Proc. of the 2000 ACM/IEEE Conf. on Supercomputing*, (2000).
14. Y. Saad. Krylov subspace methods on supercomputers. *SIAM J. of Sci. Stat. Comput.*, 10:1200–1232, (1989).
15. K. Puolamäki, M. Fortelius, and H. Mannila. Seriation in paleontological data using markov chain monte carlo methods. *PLoS Computational Bioglogy*, 2(2), (2006).
16. A. van Zuylen et al.Deterministic pivoting algorithms for constrained ranking and clustering problems. In: *ACM-SIAM SODA*, pages 405–414, (2007).