Contextual Recommendations for Groups

Kostas Stefanidis*, Nafiseh Shabib, Kjetil Nørvåg, and John Krogstie

Department of Computer and Information Science
Norwegian University of Science and Technology
Trondheim, Norway
{kstef,shabib,kjetil.norvag,krogstie}@idi.ntnu.no

Abstract. Recommendation systems have received significant attention, with most of the proposed methods focusing on recommendations for single users. Recently, there are also approaches aiming at either group or context-aware recommendations. In this paper, we address the problem of *contextual recommendations for groups*. We exploit a hierarchical context model to extend a typical recommendation model to a general context-aware one that tackles the information needs of a group. We base the computation of contextual group recommendations on a subset of preferences of the users that present the most similar behavior to the group, that is, the users with the most similar preferences to the preferences of the group members, for a specific context. This subset of preferences includes the ones with context equal to or more general than the given context.

1 Introduction

Recommendation systems provide users with suggestions about products, movies, videos, pictures and many other items. Many systems, such as Amazon, NetFlix and MovieLens, have become very popular. Typically, recommendation approaches are distinguished between: *content-based*, that recommend items similar to those the user previously preferred (e.g., [20, 15]), *collaborative filtering*, that recommend items that users with similar preferences liked (e.g., [13, 8]) and *hybrid*, that combine content-based and collaborative ones (e.g., [3, 5]).

The two types of entities that are dealt with in recommendation systems, i.e., users and items, are represented as sets of ratings, preferences or features. Assume, for example, a restaurant recommendation application (e.g., ZAGAT.com). Users initially rate a subset of restaurants that they have already visited. Ratings are expressed in the form of preference scores. Then, a recommendation engine estimates preference scores for the items, e.g., restaurants, that are not rated by a user and offers appropriate recommendations. Once the unknown scores are computed, the k items with the highest scores are recommended to the user.

* This work was carried out during the tenure of an ERCIM "Alain Bensoussan" Fellowship Programme. This Programme is supported by the Marie Curie Co-funding of Regional, National and International Programmes (COFUND) of the European Commission.

Since recommendations are typically personalized, different users are presented with different suggestions. However, there are cases in which a group of people participates in a single activity. For instance, visiting a restaurant or a tourist attraction, watching a movie or a TV program and selecting a holiday destination are examples of recommendations well suited for groups of people. For this reason, recently, there are approaches addressing the problem of identifying recommendations for groups, trying to satisfy the preferences of all the group members (e.g., [18, 2, 4, 16]).

Moreover, often users have different preferences under different circumstances. For instance, the current weather conditions may influence the place one wants to visit. For example, when it rains, a museum may be preferred over an open-air archaeological site. *Context* is a general term used in several domains, such as in machine learning and knowledge acquisition [9, 6]. Our focus here is on how context can be used in conjunction with recommendation systems. In this respect, we consider as context any information that can be used to characterize the situations of an entity, where an entity is a person, place, or object that is relevant to the interaction between a user and an application [11]. Common types of context include the computing context (e.g., network connectivity, nearby resources), the user context (e.g., profile, location), the physical context (e.g., noise levels, temperature) and time [10, 7]. Several approaches, such as [1] and [19], extend the typical recommendation systems beyond the two dimensions of users and items to include further contextual information.

In this paper, we address the problem of *contextual recommendations for groups*. In general, different approaches have been proposed in the research literature focusing on either group or context-aware recommendations. However, as far as we know, this is the first work presenting a complete model for contextual recommendations for groups. The context model of our previous work [23] serves as a building brick for extending a typical recommendation model to a general context-aware one that tackles the information needs of a group.

The computation of contextual group recommendations proceeds in the following main phases. First, given a group of users along with a context state, or situation, we locate the users that exhibit the most similar behavior or, in other words, have expressed the most similar preferences, to the group for the given context. We call such users *peers* of the group. Next, we employ the peers preferences defined for the given context to identify the items to be suggested to the group. Since many times there are no or not enough preferences for a specific context, we consider also issues underlying context relaxation and so, employ preferences with context more general than the given one.

The rest of the paper is organized as follows. Sect. 2 presents our context model, as well as our model for contextual single user and group recommendations. Sect. 3 focuses on the three main phases for computing contextual group recommendations, namely, (i) peers selection, (ii) preferences selection and (iii) recommendations computation. Finally, Sect. 4 draws conclusions and future work.

2 A Contextual Group Recommendation Model

Assume a set of items $\mathcal{I}$ and a set of users $\mathcal{U}$ interacting with a recommendation application. Each user $u \in \mathcal{U}$ may express, for a context state cs, a contextual preference for an item $i \in \mathcal{I}$, which is denoted by $cpref(u, i, cs)$ and lies in the range $[0.0, 1.0]$. As a running example, we shall use a movie recommendation application.

In the following, we first present our context model and then focus on the specification of a contextual recommendation model for single users and groups.

2.1 Context Model

A variety of models for context have been proposed (see, for example, [21] for a survey). We follow the data-centric approach of [23]. Context is modeled as a set of n context parameters $C_1, \ldots, C_n$, where each C_i, $1 \leq i \leq n$, captures information that is not part of the database, such as the user location, the current weather or time. For our movie example, let us assume that context consists of *Weather* and *Time period*. Each context parameter takes values from a hierarchical domain, so that different levels of abstraction for the captured context data are introduced.

In particular, each context parameter has multiple levels organized in a hierarchy schema. Let C be a context parameter with $m > 1$ levels, L_i, $1 \leq i \leq m$. We denote its hierarchy schema as $L = (L_1, ..., L_m)$. L_1 is called the lowest or most detailed level of the hierarchy schema and L_m the top or most general one. We define a total order among the levels of L such that $L_1 \prec \ldots \prec L_m$ and use the notation $L_i \preceq L_j$ between two levels to mean $L_i \prec L_j$ or $L_i = L_j$. Fig. 1 depicts the hierarchy schemas of the context parameters of our running example. For instance, the hierarchy schema of context parameter *Time period* has three levels: *occasion* (L_1), *interval* (L_2) and the top level *ALL* (L_3). Each level L_j, $1 \leq j \leq m$, is associated with a domain of values, denoted $dom_{L_j}(C)$. For all parameters, their top level has a single value *All*, i.e., $dom_{L_m}(C) = \{All\}$. A concept hierarchy is an instance of a hierarchy schema, where the concept hierarchy of a context parameter C with m levels is represented by a tree with m levels with nodes at each level j, $1 \leq j \leq m$, representing values in $dom_{L_j}(C)$. The root node (i.e., level m) represents the value *All*. Fig. 1 depicts the concept hierarchies of the context parameters of our running example. For instance, for the context parameter *Time period*, *holidays* is a value of level *interval*. The relationship between the values at the different levels of a concept hierarchy is achieved through the use of a family of ancestor and descendant functions [24]. Finally, we define the domain, $dom(C)$, of C as: $dom(C) = \cup_{j=1}^{m} dom_{L_j}(C)$.

A context state cs is defined as an n-tuple $(c_1, \ldots, c_n)$, where $c_i \in dom(C_i)$, $1 \leq i \leq n$. For instance, (*warm, holidays*) and (*cold, weekend*) are context states for our movie example. The set of all possible context states, called world CW, is the Cartesian product of the domains of the context parameters: $CW = dom(C_1) \times \ldots \times dom(C_n)$.

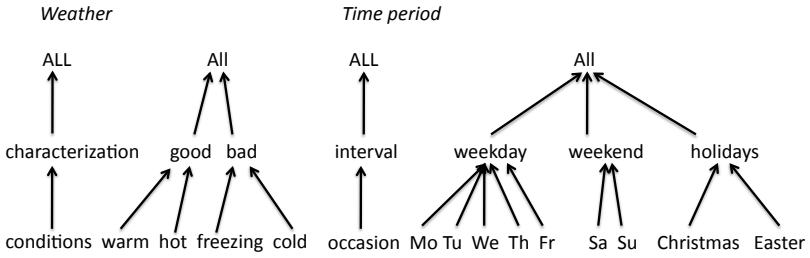

Fig. 1. Hierarchy schema and concept hierarchy of *Weather* and *Time period*.

2.2 Contextual Recommendations for Single Users

In general, there are different ways to estimate the relevance of an item for a user under a specific context state by employing a set of available user contextual preferences of the form $cpref(u, i, cs)$. The meaning of such a preference is that in the context state specified by cs, the movie, in our case, i was rated by user u with a score. For example, according to $cpef$(*Tim, The Hangover,* (*warm, weekend*)) = 0.8, *Tim* gave a high rate to the comedy movie *The Hangover* at a *warm weekend*, while $cpef$(*Alice, Toy Story,* (*cold, Christmas*)) = 0.9 defines that *Alice* likes the animation movie *Toy Story* during *cold Christmas*.

Our work falls into the *collaborative filtering* category. The key concept of collaborative filtering is to use preferences of other users that exhibit the most similar behavior to a given user, for a specific context, in order to predict relevance scores for unrated items. Similar users are located via a *similarity function* $simU_{cs}(u, u')$, that evaluates the proximity between u and u' for cs.

We use $\mathcal{P}_{u,cs}$ to denote the set of the most similar users to u for a context state cs. Or, in other words, the users with preferences similar to the preferences of u for cs. We refer to such users as the *peers* of u for cs. Several methods can be employed for selecting $\mathcal{P}_{u,cs}$. A direct method is to locate those users u' with similarity $simU_{cs}(u, u')$ greater than a threshold value. This is the method used in this work. Formally, peers are defined as follows:

Definition 1 (Peers of a Single User). *Let $\mathcal{U}$ be a set of users. The peers $\mathcal{P}_{u,cs}$, $\mathcal{P}_{u,cs} \subseteq \mathcal{U}$, of a user $u \in \mathcal{U}$, for a context state cs, is a set of users, such that, $\forall u' \in \mathcal{P}_{u,cs}$, $simU_{cs}(u, u') \geq \delta$ and $\forall u'' \in \mathcal{U} \backslash \mathcal{P}_{u,cs}$, $simU_{cs}(u, u'') < \delta$, where δ is a threshold similarity value.*

Clearly, one could argue for other ways of selecting $\mathcal{P}_{u,cs}$, for instance, by taking the k most similar users to u. Our main motivation is that we opt for selecting only highly connected users even if the resulting set of users $\mathcal{P}_{u,cs}$ is small.

Next, we define the contextual relevance of an item recommendation for a user.

Definition 2 (Single User Contextual Relevance). *Given a user $u \in \mathcal{U}$ and his peers $\mathcal{P}_{u,cs}$ for a context state cs, the single user contextual relevance of an item $i \in \mathcal{I}$ for u under cs, such that, $\nexists cpref(u,i,cs)$, is:*

$$crel(u,i,cs) = \frac{\sum_{u' \in (\mathcal{P}_{u,cs} \cap \mathcal{X}_{i,cs})} simU_{cs}(u,u') cpref(u',i,cs)}{\sum_{u' \in (\mathcal{P}_{u,cs} \cap \mathcal{X}_{i,cs})} simU_{cs}(u,u')}$$

where $\mathcal{X}_{i,cs}$ is the set of users in $\mathcal{U}$ that have expressed a preference for item i for context state cs.

2.3 Contextual Recommendations for Groups

The large majority of recommendation systems are designed to make personal recommendations, i.e., recommendations for single users. However, there are cases in which the items to be selected are not intended for personal usage but for a group of users. For example, assume a group of friends or a family that is planning to watch a movie together. Existing methods to construct a ranked list of recommendations for a group of users can be classified into two approaches [12]. The first approach aggregates the recommendations of each user into a single recommendation list (e.g., [2, 4, 17]), while the second one creates a joint profile for all users in the group and provides the group with recommendations computed with respect to this joint profile (e.g., [14, 25]).

In our work, we adopt the second approach to offer context-aware recommendations to groups. The first step towards this direction is to locate the similar users to the group, whose preferences will be used for making suggestions.

For a context state cs, we define the similarity between a user u and a group of users $\mathcal{G}$ as follows:

$$simG_{cs}(u,\mathcal{G}) = Aggr_{u' \in \mathcal{G}}(simU_{cs}(u,u'))$$

We employ two different designs regarding the aggregation method $Aggr$, each one carrying different semantics: (i) the *least misery design*, where the similarity between the user u and the group $\mathcal{G}$ is equal to the minimum similarity between u and any other user in $\mathcal{G}$, and (ii) the *fair design*, where the similarity between u and $\mathcal{G}$ is equal to the average similarity between u and all users in $\mathcal{G}$. The least misery design captures cases where strong user preferences act as a veto, e.g., do not recommend thriller movies to a group when a group member extremely dislike them, while the fair design captures more democratic cases where the majority of the group is satisfied.

Then, the peers of a group for a context state cs are defined as:

Definition 3 (Peers of a Group). *Let $\mathcal{U}$ be a set of users. The peers $\mathcal{P}_{G,cs}$, $\mathcal{P}_{G,cs} \subseteq \mathcal{U}$, of a group $\mathcal{G}$, for a context state cs, is a set of users, such that, $\forall u \in \mathcal{P}_{G,cs}$, $simG_{cs}(u,\mathcal{G}) \geq \delta'$ and $\forall u' \in \mathcal{U} \backslash \mathcal{P}_{G,cs}$, $simG_{cs}(u',\mathcal{G}) < \delta'$, where δ' is a threshold similarity value.*

Based on the notion of peers for a group, we define next the contextual relevance of an item for a group for a specific context state.

Definition 4 (Group Contextual Relevance). *Given a group $\mathcal{G}$ and its peers $\mathcal{P}_{\mathcal{G},cs}$ for a context state cs, the group contextual relevance of an item $i \in \mathcal{I}$ for $\mathcal{G}$ under cs, such that, $\forall u \in \mathcal{G}$, $\nexists cpref(u,i,cs)$, is:*

$$crel(\mathcal{G},i,cs) = \frac{\sum_{u \in (\mathcal{P}_{\mathcal{G},cs} \cap \mathcal{X}_{i,cs})} simG_{cs}(u,\mathcal{G}) cpref(u,i,cs)}{\sum_{u \in (\mathcal{P}_{\mathcal{G},cs} \cap \mathcal{X}_{i,cs})} simG_{cs}(u,\mathcal{G})}$$

where $\mathcal{X}_{i,cs}$ is the set of users in $\mathcal{U}$ that have expressed a preference for item i for context state cs.

3 Computing Contextual Group Recommendations

A high level representation of the main components of the architecture of our system is depicted in Fig. 2. First, a group poses a query presenting its information needs. Each query is enhanced with a contextual specification, that is, a context state denoted as cs^Q. The context of the query may be postulated by the application or be explicitly provided by the group as part of the query. Typically, in the first case, the context associated with a query corresponds to the current context, that is, the context surrounding the group at the time of the submission of the query. Such information may be captured using appropriate devices and mechanisms, such as temperature sensors or GPS-enabled devices for location. Besides this implicit context, a group may explicitly specify a context state. For example, assume a group that expresses an exploratory query asking for interesting movies to watch over the coming *cold weekend*.

Given a specific query, computing contextual group recommendations involves three phases: peers selection, preferences selection and recommendations computation. In following, we describe each of these phases in detail.

Peers Selection. For locating the peers of a group $\mathcal{G}$ for a context state cs, we need to calculate the similarity measures $simG_{cs}(u,\mathcal{G})$, $\forall u \in \mathcal{U} \backslash \mathcal{G}$. Those users u with similarity $simG_{cs}(u,\mathcal{G})$ greater than δ' represent the peers of $\mathcal{G}$ for cs, $\mathcal{P}_{G,cs}$.

The notion of user similarity is important, since it determines the produced peers set. We use here a simple variation; that is, we use distance instead of similarity. More specifically, we define the distance between two users as the Euclidean distance over the items rated by both under the same context state. Let $u, u' \in \mathcal{U}$ be two users, $\mathcal{I}_u$ be the set of items for which $\exists cpref(u,i,cs)$, $\forall i \in \mathcal{I}_u$, and $\mathcal{I}_{u'}$ be the set of items for which $\exists cpref(u',i,cs)$, $\forall i \in \mathcal{I}_{u'}$. We denote by $\mathcal{I}_u \cap \mathcal{I}_{u'}$ the set of items for which both users have expressed preferences for cs. Then, the distance between u, u' is defined as:

$$distU_{cs}(u,u') = \frac{\sqrt{\sum_{i \in \mathcal{I}_u \cap \mathcal{I}_{u'}} (cpref(u,i,cs) - cpref(u',i,cs))^2}}{|\mathcal{I}_u \cap \mathcal{I}_{u'}|}$$

The similarity between two users, $simU_{cs}(u,u')$, is equal to $1 - distU_{cs}(u,u')$, based on which the similarity between a user and a group is computed.

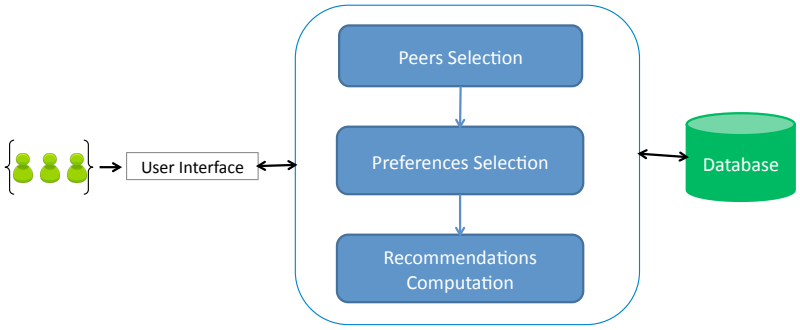

Fig. 2. System architecture.

Preferences Selection. Given a group $\mathcal{G}$, preferences selection determines which preferences from the peers of $\mathcal{G}$ will be employed for making recommendations. For example, assume that a group wants to find a movie to watch on a *Sunday*. Then, the peers preferences for *weekdays* are outside the query context.

Clearly, in terms of context, a preference $cpref(u, i, cs)$ can be used if cs is equal to the query context cs^Q. However, when there are no such peers preferences, or when their number is small, we may need to select, in addition, preferences whose context state is not necessarily the same with cs^Q, but close enough to it. To determine how close the preference and query contexts are, we rely on an appropriate distance measure. Since our context parameters take values from hierarchical domains, we exploit this fact and relate contexts expressed at different levels of detail. For instance, we can relate a context in which the parameter *Time period* is instantiated to a specific occasion (e.g., *Christmas*) with a context in which the same parameter describes a more general period (e.g., *holidays*). Intuitively, a preference defined for a more general value, e.g., *holidays*, may be considered applicable to a query about a more specific one, e.g., *Christmas*. In general, we relate the context of a preference to the context of a query, if the first one is more general than the second, that is, if the context values specified in cs are equal to or more general than the ones specified in cs^Q. In this case, we say that the preference context *covers* the query context [23].

Given a preference state $cs = (c_1, \ldots, c_n)$ and a query state $cs^Q = (c_1^Q, \ldots, c_n^Q)$, where cs covers cs^Q, we quantify their relevance based on how far away are their values in the corresponding hierarchies.

$$dist_H(cs, cs^Q) = \sum_{i=1}^{n} d_H(level(c_i),\ level(c_i^Q)),$$

where $level(c_i)$ (resp. $level(c_i^Q)$) is the hierarchy level of value c_i (resp. c_i^Q) of parameter C_i and d_H is equal to the number of edges that connect $level(c_i)$ and $level(c_i^Q)$ in the hierarchy of C_i, $1 \leq i \leq n$.

[22] studies different ways of relaxing context. In particular, a context parameter can be relaxed *upwards* by replacing its value by a more general one, *downwards* by replacing its value by a set of more specific ones or *sideways* by replacing its value by sibling values in the hierarchy.

The output of this phase is a set of m preferences with contexts closest to the query context, sorted on the basis of their distance to cs^Q, from the set of preferences of the users in $\mathcal{P}_{G,cs}$.

Recommandations Computation. For estimating the value of an item i for a group $\mathcal{G}$ under a context state cs, we compute its group contextual relevance $crel(\mathcal{G}, i, cs)$ (Def. 4), taking into account the output of the previous phase. We do not compute scores for all items in $\mathcal{I}$, but only for the items $\mathcal{I}'$, $\mathcal{I}' \subseteq \mathcal{I}$, that satisfy the selection conditions of the posed query. As a post-processing step, we rank the items in $\mathcal{I}'$ on the basis of their score and report the k items with the highest scores.

4 Conclusions

The focus of this paper is on contextual recommendations for groups. Context is modeled using a set of context parameters that take values from hierarchical domains. A context state corresponds to an assignment of values to each of the context parameters from its corresponding domain. User preferences are augmented with context states that specify the situations under which preferences hold. Given a group of users associated with a context state, we consider the problem of providing the group with context-aware recommendations. To do this, we follow a collaborative filtering approach that uses the preferences of the similar users to the group members defined for the context surrounding the group at the time of recommendations computation or any other explicitly defined context.

In our current work, we are developing a Java prototype to add a capability for producing contextual group recommendations to our group recommendations system [17]. There are many directions for future work. One is to extend our model so as to support additional ways for locating the peers of a group of users. Another direction for future work is to consider recency issues when computing contextual recommendations. For example, since usually the most recent user preferences reflect better the current trends, it is promising to examine if they should contribute more in the computation of the contextual group recommendations.

References

1. G. Adomavicius, R. Sankaranarayanan, S. Sen, and A. Tuzhilin. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Trans. Inf. Syst.*, 23(1):103–145, 2005.

2. S. Amer-Yahia, S. B. Roy, A. Chawla, G. Das, and C. Yu. Group recommendation: Semantics and efficiency. *PVLDB*, 2(1):754–765, 2009.
3. M. Balabanovic and Y. Shoham. Content-based, collaborative recommendation. *Commun. ACM*, 40(3):66–72, 1997.
4. L. Baltrunas, T. Makcinskas, and F. Ricci. Group recommendations with rank aggregation and collaborative filtering. In *RecSys*, pages 119–126, 2010.
5. C. Basu, H. Hirsh, and W. W. Cohen. Recommendation as classification: Using social and content-based information in recommendation. In *AAAI/IAAI*, pages 714–720, 1998.
6. C. Bolchini, C. Curino, G. Orsi, E. Quintarelli, R. Rossato, F. A. Schreiber, and L. Tanca. And what can context do for data? *Commun. ACM*, 52(11):136–140, 2009.
7. C. Bolchini, C. Curino, E. Quintarelli, F. A. Schreiber, and L. Tanca. A data-oriented survey of context models. *SIGMOD Rec.*, 36(4):19–26, 2007.
8. J. S. Breese, D. Heckerman, and C. M. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *UAI*, pages 43–52, 1998.
9. P. Brézillon. Context in artificial intelligence: I. a survey of the literature. *Computers and Artificial Intelligence*, 18(4), 1999.
10. G. Chen and D. Kotz. A Survey of Context-Aware Mobile Computing Research. Technical Report TR2000-381, Dartmouth College, Computer Science, November 2000.
11. A. K. Dey. Understanding and using context. *Personal Ubiquitous Comput.*, 5(1):4–7, 2001.
12. A. Jameson and B. Smyth. Recommendation to groups. In *The Adaptive Web*, pages 596–627, 2007.
13. J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl. Grouplens: Applying collaborative filtering to usenet news. *Commun. ACM*, 40(3):77–87, 1997.
14. J. F. McCarthy and T. D. Anagnost. Musicfx: an arbiter of group preferences for computer supported collaborative workouts. In *CSCW*, page 348, 2000.
15. R. J. Mooney and L. Roy. Content-based book recommending using learning for text categorization. In *ACM DL*, pages 195–204, 2000.
16. I. Ntoutsi, K. Stefanidis, K. Nørvåg, and H.-P. Kriegel. Fast group recommendations by applying user clustering. In *ER*, 2012.
17. I. Ntoutsi, K. Stefanidis, K. Nørvåg, and H.-P. Kriegel. grecs: A group recommendation system based on user clustering (demo paper). In *DASFAA (2)*, pages 299–303, 2012.
18. M. O'Connor, D. Cosley, J. A. Konstan, and J. Riedl. Polylens: A recommender system for groups of user. In *ECSCW*, pages 199–218, 2001.
19. C. Palmisano, A. Tuzhilin, and M. Gorgoglione. Using context to improve predictive modeling of customers in personalization applications. *IEEE Trans. Knowl. Data Eng.*, 20(11):1535–1549, 2008.
20. M. J. Pazzani and D. Billsus. Learning and revising user profiles: The identification of interesting web sites. *Machine Learning*, 27(3):313–331, 1997.
21. K. Stefanidis, G. Koutrika, and E. Pitoura. A survey on representation, composition and application of preferences in database systems. *ACM Trans. Database Syst.*, 36(3):19, 2011.
22. K. Stefanidis, E. Pitoura, and P. Vassiliadis. On relaxing contextual preference queries. In *MDM*, pages 289–293, 2007.
23. K. Stefanidis, E. Pitoura, and P. Vassiliadis. Managing contextual preferences. *Inf. Syst.*, 36(8):1158–1180, 2011.

24. P. Vassiliadis and S. Skiadopoulos. Modelling and optimisation issues for multidimensional databases. In *CAiSE*, pages 482–497, 2000.
25. Z. Yu, X. Zhou, Y. Hao, and J. Gu. Tv program recommendation for multiple viewers based on user profile merging. *User Model. User-Adapt. Interact.*, 16(1):63–82, 2006.