Diversity over Continuous Data

Marina Drosou
Computer Science Department
University of Ioannina, Greece
mdrosou@cs.uoi.gr

Evaggelia Pitoura
Computer Science Department
University of Ioannina, Greece
pitoura@cs.uoi.gr

Abstract

Result diversification has recently attracted much attention as a means of increasing user satisfaction in recommendation systems and web search. In this work, we focus on achieving content diversity in the case of continuous data delivery, such as in the context of publish/subscribe systems. We define sliding-window diversity and present a suite of heuristics for its efficient computation along with some performance results.

1 Introduction

With the explosion of the amount of information currently available online, publish/subscribe systems offer an attractive alternative to searching by providing a proactive model of information supply. In such systems, users express their interest in specific pieces of data (or events) via subscriptions. Then, they are notified whenever some other user generates (or publishes) an event that matches one of their subscriptions. Typically, all subscriptions are considered equally important and users are notified whenever a published event matches any of their subscriptions. However, getting notified about all matching events may lead to overwhelming the users with large amounts of notifications, thus hurting the acceptability of publish/subscribe systems.

Today, most user searches have an exploratory nature, in the sense that users are mostly interested in retrieving various information about their search topic. Therefore, recently, *result diversification* has attracted considerable attention as a means of enhancing user satisfaction in recommender systems and web search (e.g. [18, 14]). We argue that diversification could also be employed in the context of publish/subscribe systems to improve the overall quality of notifications delivered to users.

In this paper, we tackle the problem of selecting k diverse information pieces (or *items*) among the information content being forwarded to users and delivering only these items to the users instead of overwhelming them with all relevant information. Diverse items may be defined in three different ways, namely in terms of (i) *novelty*, i.e. choosing to deliver items that contain new information when compared to previously delivered ones (e.g. [5, 17]), (ii) *coverage*, i.e. choosing to deliver items that belong to different categories (e.g. [3]) and (iii) *content* (or *similarity*), i.e. choosing to deliver items that are dissimilar to each other (e.g. [16]). Motivated by the fact that publish/subscribe systems are simultaneously used by many different publishing sources that often publish overlapping information, in this work, we focus on content diversity among the items (i.e. events)

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

of a publish/subscribe system and seek to process the continuous flow of information in such a way as to locate and deliver to users events that are distant (dissimilar) to each other, given a budget of k.

In this paper, we introduce the problem of diversity over continuous data, present initial algorithms and suggest issues for further research. The rest of this paper is structured as follows. In Section 2, we define the k-diversity problem and present efficient solutions, while in Section 3, we adapt these solutions for continuous data. Section 4 focuses on the combination of multiple criteria, namely diversity and relevance, for the final ranking of information. Section 5 briefly reviews related work and, finally, Section 6 concludes this paper.

2 Content Diversity

There are various forms of diversity. Here, we focus on content diversity, so that the users receive dissimilar content. Specifically, given a set of n items, we aim at selecting k items out of them, such that, the average pairwise distance between the selected items is maximized. More formally:

Definition 1: Let $P = \{p_1, \ldots, p_n\}$ be a set of n items and k an integer with $k \leq n$. Let also $d(p_i, p_j)$, $p_i, p_j \in P$, be a distance metric between items p_i and p_j. The k-diversity problem is to locate a subset S^* of P, such that:

$$S^* = \underset{\substack{S \subseteq P \\ |S|=k}}{\operatorname{argmax}} f_D(S), \text{ where } f_D(S) = \frac{1}{k(k-1)} \sum_{i=1}^{k} \sum_{j>i}^{k} d(p_i, p_j) \text{ and } p_i, p_j \in S \quad (1)$$

The problem of selecting the k items having the maximum average pairwise distance out of n items is similar to the *p-dispersion* problem. This problem, as well as a number of its variations (e.g. select p out of n items so that the minimum distance between any two of them is maximized), have been extensively studied in operations research and are in general known to be NP-hard [8, 9]. Thus, to solve large instances of the problem, we need to rely on heuristics. A number of heuristics have been proposed in the literature (e.g. [9]), ranging from applying exhaustive algorithms to adapting traditional optimization techniques. A general issue that hinders the development of efficient incremental solutions to the problem is that the $k-1$ most diverse items of a set P are not necessarily a subset of its k most diverse items. For example, consider as items the points on the circumstance of a circle and their euclidean distances. The two furthest apart points are (any) two antidiametric ones. However, no antidiametric points belong to the three most diverse points.

There are two families of heuristics that locate good solutions for the k-diversity problem at a reasonable time: *greedy* heuristics and *interchange* heuristics (the reader is referred to [6] for a study of various heuristics and their behavior). Greedy heuristics make use of two sets: the initial set P and a set S which will eventually contain the selected items. Items are iteratively moved from P to S and vice versa until $|S| = k$ and $|P| = n-k$. There are two main variations. In the *Greedy Construction* heuristic, initially, $|P| = n$ and $|S| = 0$. First, the two furthest apart items of P are added to S. Then, at each iteration, one more item is added to S. The item that is added is the one that has the maximum item-set distance from S. We define the *item-set distance* $setdist(p_i, S)$ between an item p_i and a set of items S as the average distance between p_i and the items in S, that is:

$$setdist(p_i, S) = \frac{1}{|S|} \sum_{p_j \in S} d(p_i, p_j) \quad (2)$$

In the *Greedy Deletion* heuristic, initially, $|P| = 0$ and $|S| = n$. At each iteration, the two closest items of S are located. One of them is then moved to P. The choice is based on the minimum item-set distance from the remaining items of S.

Generally, the Greedy Construction heuristic (denoted GC, Algorithm 1) performs better, both in terms of the achieved diversity as well as in terms of execution time. Therefore, we will consider it further in this work.

Algorithm 1 Greedy Construction Heuristic (GC).

Input: The initial set of items P, the number of wanted items k and a threshold r.
Output: The set S with the k most diverse items.

1: Set R to be a random subset of P with $|R| = r$
2: find p_1, p_2, s.t. $d(p_1, p_2) = \max\{d(p_1, p_2) : p_1, p_2 \in R, p_1 \neq p_2\}$
3: $S \leftarrow \{p_1, p_2\}$
4: **while** $|S| < k$ **do**
5: find $p_i \in P \backslash S$, s.t. $setdist(p_i, S) = \max\{setdist(p_j, S) : p_j \in P \backslash S\}$
6: $S \leftarrow S \cup \{p_i\}$
7: **end while**
8: **return** S

Algorithm 2 First Pairwise Interchange Heuristic (FI).

Input: The initial set of items P, the number of wanted items k and a threshold h.
Output: The set S with the k most diverse items.

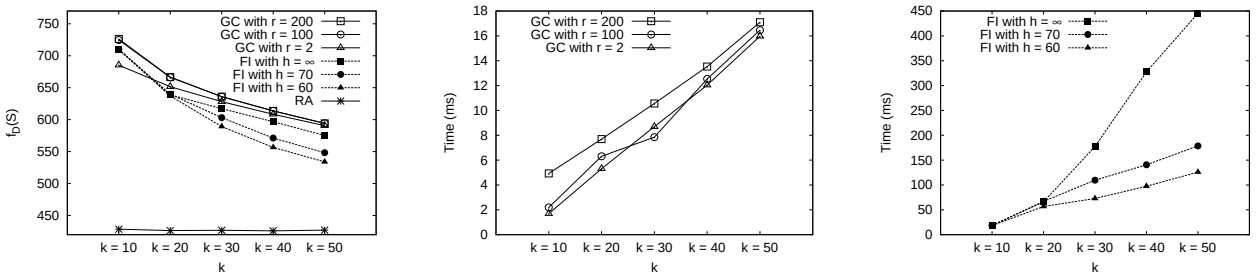
1: Set S to be a random solution
2: **while** less than h iterations have been performed **do**
3: find p_1, p_2, s.t. $d(p_1, p_2) = \min\{d(p_1, p_2) : p_1, p_2 \in S, p_1 \neq p_2\}$
4: **for all** $p_i \in P \backslash S$ **do**
5: $S' \leftarrow \{S \backslash \{p_1\}\} \cup \{p_i\}$
6: $S'' \leftarrow \{S \backslash \{p_2\}\} \cup \{p_i\}$
7: **if** $f(S') > f(S)$ and $f(S') \geq f(S'')$ **then**
8: $S \leftarrow S'$; **break**
9: **end if**
10: **if** $f(S'') > f(S)$ and $f(S'') > f(S')$ **then**
11: $S \leftarrow S''$; **break**
12: **end if**
13: **end for**
14: **end while**
15: **return** S

The complexity of GC is $O(n^2)$. However, this is due to the first step of the algorithm where the two furthest apart items have to be located. The rest of the algorithm takes $O(k^2 n)$ time. Therefore, to reduce the distance computations required by GC, we opt to initialize S by selecting the two furthest apart items from a randomly selected subset of P with size equal to r, $r < n$, instead of using the whole set.

Interchange heuristics are initialized with a random solution S and then iteratively attempt to improve that solution by interchanging an item in the solution with another item that is not in the solution. The item that is eliminated from the solution at each iteration is one of the two closest items in it. Again, there are two main variations. The *First Pairwise Interchange* heuristic (denoted FI) performs at each iteration the first interchange that improves the solution, while the *Best Pairwise Interchange* heuristic (denoted BI) considers all possible interchanges and performs the one that improves the solution the most.

None of the two algorithms clearly outperforms the other, while their worst case complexity is $O(n^k)$. Each distinct iteration of FI is on average faster than an iteration of BI. However, FI is more likely to perform more iterations. Even though there is no clear winner in terms of execution time, FI usually locates better solutions [9, 6]. This is the reason we will focus on this variation (Algorithm 2) in the rest of this work. There are two ways to limit the iterations performed by FI. We can either set a bound, say h, on the maximum number of possible interchanges to be performed or allow interchanges to continue as long as the solution improves by a given threshold. In this paper, we choose to directly control the number of iterations, so that we can get reasonable execution times.

Next, we provide experimental results to evaluate the performance of these heuristics, both in terms of efficiency (execution time) as well as effectiveness (achieved diversity). Our goal is not to provide a comprehensive evaluation of the two heuristics but to present a couple of experiments to provide some insight about their performance. We create synthetic datasets consisting of 200 data points in the euclidean space, where the values of each dimension are in $[0, 1000]$ and use as distance d the euclidean distance. The heuristics were implemented in JDK 6 and run on a Windows XP PC with a 2.4 GHz Intel Core 2 Processor and 2.00 GB of RAM. In the following, we show results averaged over 500 runs for 5-dimensional data items drawn from a normal distribution. Figure 1 shows results for various values of k for the two heuristics. For GC, we use $r = 200$ (i.e. initializing the heuristic with the whole set P), $r = 100$ and $r = 2$ (i.e. initializing the heuristic with two random items), while for FI, we either allow a certain number of iterations ($h = 60$ or $h = 70$) or allow the algorithm to run until



(a) Achieved diversity for GC and FI. (b) Execution time for GC. (c) Execution time for FI.

Figure 1: Average achieved diversity and execution time for GC and FI.

convergence ($h = \infty$). We also report the average diversity of k randomly selected items (denoted RA). Both heuristics achieve comparable diversity with GC slightly outperforming FI. GC is much faster that FI. Also, note that reducing the initial items r considered by GC does not affect the achieved diversity considerably.

3 Content Diversity in Publish/Subscribe Systems

Publish/subscribe systems provide an alternative way of receiving data of interest. In such systems, users express their interests in items through subscriptions. New items (or events) published by an information source (or publisher) are matched against the subscriptions and those items that match subscriptions are delivered to the corresponding users. Some examples of publish/subscribe systems or proactive delivery include news alerts, RSS feeds and notification services in social networks. As with web search, user subscriptions are often exploratory in nature, in the sense that users do not really know what they want and may not be precise on expressing their information needs. Thus, recent works have suggested that event matching should also be best effort [13, 19]. For this reason, to further enhance user satisfaction, we consider the case where not all matching items are forwarded to the subscribers but just the k most diverse of them.

This is an instance of continuous data delivery. Since events are published and matched in a continuous manner, we need to define over which subsets of data we apply diversification. In our previous work [7], we have considered three fundamental models for item delivery: (i) *periodic*, (ii) *sliding-window* and (iii) *history-based filtering* delivery. Given a budget of k, with periodic delivery, the k most diverse items are computed over disjoint periods of length T and are forwarded to the subscribers at the end of the period. With sliding-window delivery, the k most diverse items are computed over sliding windows of length w, so that, an item is forwarded, if and only if, it is part of the k most diverse items in any of the windows it belongs to. Finally, history-based filtering forwards new items as they are matched, if and only if, they are dissimilar enough to the k most diverse items recently delivered. The lengths T and w can be defined either in time units (e.g. as "the 10 most diverse items matched per hour" and, respectively, "the 10 most diverse items matched in the last hour") or in number of items (e.g. as "the 10 most diverse items per 100 matched ones" and, respectively, "the 10 most diverse items among the 100 most recently matched ones").

Here, we allow windows not only to slide but also to "jump", i.e. move forward more than one position in the stream of matching items each time. We call these windows *jumping windows* (Figure 2). Assuming windows of length w and a jump step of length j, with $j < w$, consequent windows overlap and share $w - j$ common items. Note that, for $w = 1$, we get sliding-window delivery, while for $w = T$, we get periodic delivery. We denote the i^{th} window as W_i and write $W_i = \{p_1, \ldots, p_w\}$, where $p_1, \ldots, p_w$ are the items that belong to W_i. Next, we define the k-diversity problem over continuous data:

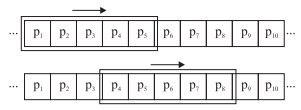


Figure 2: A jumping window with $w = 5$ and $j = 3$.

Definition 2: Let $\mathbb{W}$ be a stream of items and consider a window sliding over $\mathbb{W}$. The sliding-window k-diversity problem is the following: In each window W_i of $\mathbb{W}$, locate and forward to the user a set S_i^*, such that:

$$S_i^* = \underset{\substack{S_i \subseteq W_i \\ |S_i|=k}}{\operatorname{argmax}} f_D(S_i) \tag{3}$$

One difficulty of computing diverse items over continuous data is that it cannot be done incrementally. Let S and S' be two sets that differ at only one item. Then, the k most diverse items of S are in general completely different than the k most diverse items of S'. As a simple example, consider 2-dimensional points in the euclidean space and the sets $S = \{(0,0),(3,3),(5,6),(1,7)\}$ and $S' = \{(3,3),(5,6),(1,7),(4,4)\}$. Then, the 2 most diverse items of S are $(0,0)$ and $(5,6)$, while the 2 most diverse items of S' are $(1,7)$ and $(3,3)$.

The straightforward solution to the problem is to re-compute the k most diverse items at each window. To do this, we will consider the GC heuristic, since, as we showed in Section 3, it consistently outperforms FI, especially in terms of execution time, which is crucial in real-time systems such as publish/subscribe. We also consider the following variation. After each window jump, some of the previous k most diverse items (say m of them) leave the window. Therefore, we initialize the GC algorithm with the $k-m$ remaining most diverse items from the previous window and then let the algorithm add m items from the new window based on their item-set distances. We denote this variation as SGC.

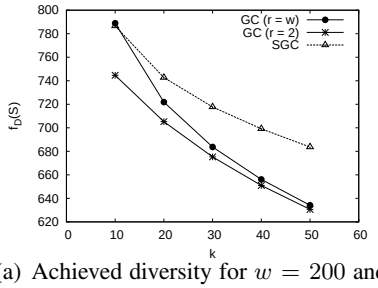
Next, we evaluate the performance of GC versus SGC. We examine GC for $r = w$ and $r = 2$. We use a stream of 10000 5-dimensional data points drawn from a normal distribution and vary k, w and j. In Figure 3, we report the average diversity achieved in each window of the stream and the corresponding average execution time. We observe that SGC constantly behaves better than both versions of GC, both in terms of diversity and time. The achieved diversity of both heuristics decreases for larger values of k and smaller values of w, as expected. SGC performs better as k increases (or w decreases), since there are more diverse items from the previous window that still remain in the new one. For the same reason, a large jump step causes the performance of SGC to degrade. The behavior of GC does not depend on the length of the jump, as the k most diverse items are recomputed at every window.

4 Content Diversity and Relevance

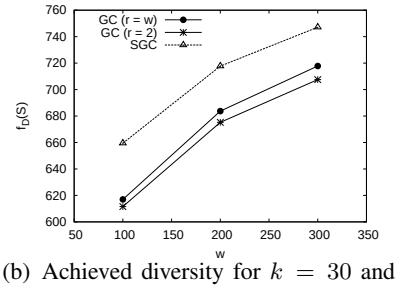
Often, diversity is just one of the quality aspects in information delivery. In general, items are also ranked based on their relevance to a user query or subscription. Besides relevance, the ranking of an item may be an indicator of its importance associated, for example, with the authority of the publisher or with the specific preferences or interests of the user. We assume that this ranking is expressed using a numeric value (or score) associated with each item p_i through some ranking function $f_R(p_i)$ that takes values in [0,1]. Then, the score of a set of items is the average score of its members.

This creates a multi-criteria problem, since we want to deliver the set of k items that are both highly preferred and diverse. There are basically two approaches to combine the two criteria, i.e. diversity and relevance: (i) employing a weighted combination of the two criteria through a weight σ and (ii) a threshold variation where we seek to maximize diversity given a minimum required score for the selected set of items (or, the dual, maximize the score of the selected items given a minimum required diversity). Placing a threshold on diversity however may be hard, since it requires an estimation of the achievable diversity.

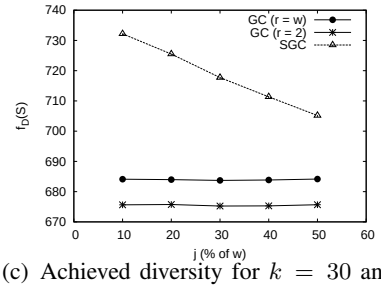
In this work, we focus on a weighted combination of relevance and diversity. Assuming that items of interest take values from a non-infinite domain, we use M to denote the maximum possible distance between any two items. Then, we redefine the k-diversity problem as locating a subset S^* of P, such that:



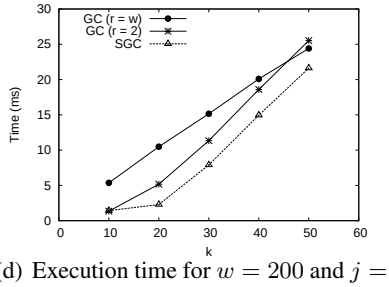
(a) Achieved diversity for $w = 200$ and $j = w \cdot 30\%$.



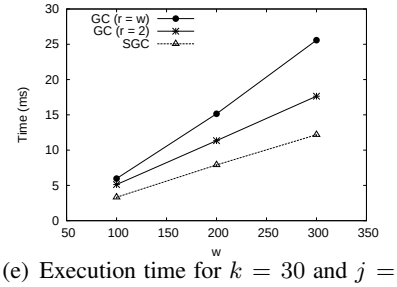
(b) Achieved diversity for $k = 30$ and $j = w \cdot 30\%$.



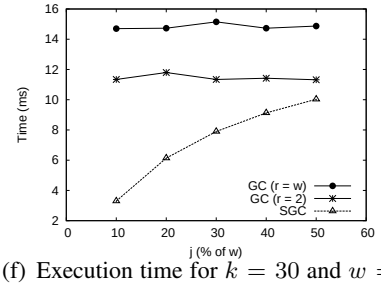
(c) Achieved diversity for $k = 30$ and $w = 200$.



(d) Execution time for $w = 200$ and $j = w \cdot 30\%$.



(e) Execution time for $k = 30$ and $j = w \cdot 30\%$.



(f) Execution time for $k = 30$ and $w = 200$.

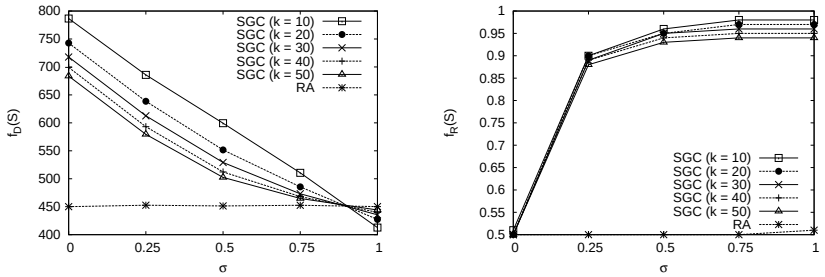
Figure 3: Average achieved diversity and execution time for the GC and SGC heuristics.

$$S^* = \underset{\substack{S \subseteq P \\ |S|=k}}{\operatorname{argmax}} g(S), \text{ where } g(S) = \sigma \cdot \frac{1}{|S|} \sum_{p_i \in S} f_R(p_i) + (1-\sigma) \cdot \frac{1}{M} f_D(S), \text{ with } \sigma \in [0,1] \tag{4}$$

Many interesting questions arise concerning the best way to combine the two measures, such as, what is the best value for the weight and whether we can adjust it dynamically based on the dataset. Here, we assume that items are associated with uniformly distributed scores. Also, the item-set distance $setdist(p_i, S)$ between an item p_i and a set of items S (Equation 2) used by SCG is re-defined accordingly to Equation 4 to consider both measures. In Figure 4, we show the average archived diversity and relevance for the windows of our stream of 10000 data items when varying σ for the SGC heuristic and for the various values of k. Figure 4 also shows the corresponding results when k random items are chosen in each window (denoted RA). Naturally, by varying the value of σ, we can tune the trade-off between the achieved diversity and relevance. We notice that when choosing items based solely on relevance ($\sigma = 1.00$), the achieved diversity drops below that of the random case (RA). In this setting, we observe a great increase in relevance as σ increases, even for $\sigma = 0.25$, while the reduction of diversity is more gradual. In this case, there is no reason to use $\sigma > 0.50$ since this results in further reduction of diversity without an analogous gain of relevance.

5 Related Work

Although the combination of diversity and relevance has attracted considerable attention recently, it is not a new concept. [4] proposed a linear combination of the two measures back in 1998, similar to the approaches more recently followed in [18, 7]. In [18], a method for topic diversification is proposed for recommendations based on the intra-list similarity, a permutation-insensitive metric introduced to assess the topical diversity of a given recommendation list. The proposed algorithm also considers the relevance of the candidate items when creating recommendation lists. [7] applies the concept of ranking based on both relevance and diversity in the context of publish/subscribe systems. The notion of diversity is also explored in database systems. Motivated



(a) Achieved diversity. (b) Achieved relevance.

Figure 4: Average achieved diversity and relevance for various values of σ.

by the fact that some database relation attributes are more important to the user, [14] proposes a method where recommendation lists consisting of database tuples are diversified by first varying the values of higher priority attributes before varying the values of lower priority ones. When the tuples are associated with scores, a scored variation of the method picks more relevant to the query tuples first. [12] tackles the problem of automatically extracting a set of features from the items of interest that is capable of differentiating the items from each other. [16] formulates the k-diversity problem as an optimization problem. Given the $n \times n$ distance matrix of the candidate items, the goal is to find a binary vector of size n that represents which items belong to the most diverse subset of size k. This is a binary problem which has to be relaxed to a real-valued problem to be solved. Recently, [10] defined a set of natural, intuitive axioms that a diversification system is expected to satisfy. The authors prove that all axioms cannot hold simultaneously and show which of them are satisfied by three main diversification methods.

Besides using content diversity, there is also related work based on different definitions of diversity. [15] proposes computing diversity based on the notion of explanations. The explanation of a given item for a user is the set of similar items the user has rated in the past. Distances between two items are measured based on their corresponding explanations. [11] proposes algorithms to capture diverse concepts in text documents, aiming at retrieving diverse sentences that can be used as snippets from search engines. These algorithms are based on coverage, i.e. how many terms of the document appear in the selected sentences, and orthogonality, i.e. how much the terms appearing in such a sentence differ from those appearing in the rest. Coverage is also considered in [3], which aims at locating diverse documents to answer a user query. Given a taxonomy of topics and a probability distribution for topics relevant to the query, the k-diversity problem is formulated as the location of a set of documents with cardinality k that maximizes the probability of as many topics as possible being represented by that set. In [5], a distinction is made between novelty, i.e. avoiding redundancy, and diversity, i.e. resolving ambiguity. Both user queries and documents are broken down into small pieces of information, called nuggets. Then, diverse documents are retrieved based on a probabilistic model of nuggets being contained in the various queries and documents.

6 Conclusions

Deriving efficient and effective algorithms for content diversity, let alone continuous content diversity, is still an open problem; we believe as broad as clustering. This paper attempts to serve as one step towards exploring some of the issues involved. To this end, we have focused on two intuitive heuristic solutions to the problem, namely a greedy and an interchange one, and their variants. Information filtering as in publish/subscribe is an instance where the need for continuous diversity arises. Publish/subscribe delivery also raises issues related to distributed data management, since both the publishers of items and the subscribers are distributed. In addition, for performance and reliability, the matching of subscriptions and publications is often done distributed. We

have implemented diversity and ranking based on user preferences in our prototype, called PrefSIENA [1], that extends SIENA [2], a popular publish/subscribe system. We are currently working on distribution issues.

References

[1] PrefSIENA. http://www.cs.uoi.gr/∼mdrosou/PrefSIENA.

[2] SIENA. http://serl.cs.colorado.edu/∼serl/dot/siena.html.

[3] R. Agrawal, S. Gollapudi, A. Halverson, and S. Ieong. Diversifying search results. In *WSDM*, 2009.

[4] J. G. Carbonell and J. Goldstein. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *SIGIR*, 1998.

[5] C. L. A. Clarke, M. Kolla, G. V. Cormack, O. Vechtomova, A. Ashkan, S. Büttcher, and I. MacKinnon. Novelty and diversity in information retrieval evaluation. In *SIGIR*, 2008.

[6] M. Drosou and E. Pitoura. Comparing diversity heuristics, Technical Report 2009-05. Computer Science Department, University of Ioannina, 2009.

[7] M. Drosou, K. Stefanidis, and E. Pitoura. Preference-aware publish/subscribe delivery with diversity. In *DEBS*, 2009.

[8] E. Erkut. The discrete *p*-dispersion problem. *European Journal of Operational Research*, 46(1), 1990.

[9] E. Erkut, Y. Ülküsal, and O. Yeniçerioglu. A comparison of *p*-dispersion heuristics. *Computers & OR*, 21(10), 1994.

[10] S. Gollapudi and A. Sharma. An axiomatic approach for result diversification. In *WWW*, 2009.

[11] K. Liu, E. Terzi, and T. Grandison. Highlighting diverse concepts in documents. In *SDM*, 2009.

[12] Z. Liu, P. Sun, and Y. Chen. Structured search result differentiation. *PVLDB*, 2(1), 2009.

[13] A. Machanavajjhala, E. Vee, M. N. Garofalakis, and J. Shanmugasundaram. Scalable ranked publish/subscribe. *PVLDB*, 1(1), 2008.

[14] E. Vee, U. Srivastava, J. Shanmugasundaram, P. Bhat, and S. Amer-Yahia. Efficient computation of diverse query results. In *ICDE*, 2008.

[15] C. Yu, L. V. S. Lakshmanan, and S. Amer-Yahia. It takes variety to make a world: diversification in recommender systems. In *EDBT*, 2009.

[16] M. Zhang and N. Hurley. Avoiding monotony: improving the diversity of recommendation lists. In *RecSys*, 2008.

[17] Y. Zhang, J. P. Callan, and T. P. Minka. Novelty and redundancy detection in adaptive filtering. In *SIGIR*, 2002.

[18] C.-N. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen. Improving recommendation lists through topic diversification. In *WWW*, 2005.

[19] C. Zimmer, C. Tryfonopoulos, and G. Weikum. MinervaDL: An architecture for information retrieval and filtering in distributed digital libraries. In *ECDL*, 2007.