Artif Intell Rev (2007) 28:227–255
DOI 10.1007/s10462-009-9100-2

P2P case storage and retrieval with an unspecified ontology

Shlomo Berkovsky · Tsvi Kuflik · Francesco Ricci

Published online: 1 April 2009

Abstract Traditional approaches for similarity-based retrieval of structured data, such as Case-Based Reasoning (CBR), have been largely implemented using centralized storage systems. In such systems, when the cases contain both numeric and free-text attributes, similarity-based retrieval cannot exploit standard speedup techniques based on multi-dimensional indexing, and the retrieval is implemented by an exhaustive comparison of the case to be solved with the whole set of stored cases. In this work, we review current research on Peer-to-Peer (P2P) and distributed CBR techniques and propose a novel approach for storage of the case-base in a decentralized Peer-to-Peer environment using the notion of Unspecified Ontology to improve the performance of the case retrieval stage and build CBR systems that can scale up to large case-bases. We develop an algorithm for efficient retrieval of approximated most-similar cases, which exploits inherent characteristics of the unspecified ontology in order to improve the performance of the case retrieval stage in the CBR problem solving cycle. The experiments show that the algorithm successfully retrieves cases close to the most-similar cases, while reducing the number of cases to be compared. Hence, it improves the performance of the retrieval stage. Moreover, the distributed nature of our approach eliminates the computational bottleneck and single point of failure of the centralized storage systems.

Keywords Case-Based Reasoning · Similarity-based retrieval · Peer-to-Peer · Unspecified ontology

S. Berkovsky (✉)
CSIRO, ICT Centre, Hobart, Australia
e-mail: shlomo.berkovsky@csiro.au

T. Kuflik
Management Information Systems Department, University of Haifa, Haifa, Israel
e-mail: tsvikak@is.haifa.ac.il

F. Ricci
Faculty of Computer Science, Free University of Bozen-Bolzano, Bolzano, Italy
e-mail: fricci@unibz.it

1 Introduction

Peer-to-Peer (P2P) computing refers to a subclass of distributed computing, where system's functionality is achieved in a fully decentralized way by using a set of distributed resources (Androutsellis-Theotokis and Spinellis 2004). P2P systems usually lack a dedicated centrally managed infrastructure, depending rather on a contribution of resources (e.g., computing power, data, and network traffic) by the connected peers. Hence, P2P systems provide pure distributed communication middleware with theoretically unlimited storage, communication, and processing capabilities. P2P systems are typically characterized by the following advantages: cost sharing and reduction, improved scalability, reliability and robustness, resource aggregation and operability, increased autonomy, dynamism, and high levels of anonymity and privacy (Milojicic et al. 2002). Due to these advantages, P2P-based approaches to knowledge management and problem solving have recently become rather popular (Androutsellis-Theotokis and Spinellis 2004; Bonifacio et al. 2003). In particular, distributed Case-Based Reasoning (CBR) models have been proposed as an effective way to deal with multiple distributed sources of knowledge (McGinty and Smyth 2001). Here, the main concern is to allow a distributed storage of knowledge (cases) and still perform various problem solving tasks (e.g., diagnosis) in an effective and efficient way by distributing and coordinating the problem solving activity on the connected peers. The peers are often modeled as agents and multi-agents technologies are also exploited (Wooldridge 2002).

After reviewing P2P technologies and distributed CBR technologies, this paper proposes to use the storage resources and the fast and robust access methods of P2P networks as a CBR component. Such a component can be used for the purpose of distributing the storage of cases and supporting distributed retrieval of similar cases. Since P2P systems do not require central management, the cases are inserted autonomously by a community of users and may be described by a highly dynamic set of terms. To maintain proper functionality, the system should eventually support users using different terms for describing similar cases or even the same case. This implies that similar cases might differ not only in the values of certain features like in classical CBR approaches, but also in the way they are described, i.e., different features are used for the same concept. Thus, efficient management of the case-base requires a stable semantic-based infrastructure allowing identification of similarities between the heterogeneously described cases. Moreover, this infrastructure should support a retrieval process that: (i) performs decentralized retrieval with low communicational overheads, and (ii) guarantees fast discovery of the most similar cases, or at least a reasonably good approximation ('reasonably good' will be defined later).

We used the hypercube-based approach of UNSO (Ben-Asher and Berkovsky 2006) for implementing P2P storage of a case-base and developed an approximated retrieval algorithm aimed at reducing the computational and communicational overheads in comparison with the traditional exhaustive retrieval. The algorithm is based on the observation that UNSO implicitly groups similar data objects, such that they are located in relatively nearby locations with respect to the P2P overlay network topology. Hence, an approximated CBR case retrieval over UNSO is performed through an expanding search starting at the location, to which the target case would have been inserted in the underlying network. This means that the cases most similar to the target case are not searched in the whole case-base, but rather the retrieval focuses on the cases that are nearby to the location of the target case. This way, the number of the compared candidate cases is reduced with respect to the traditional retrieval exhaustively comparing the target case with all the cases stored in the case-base. Since UNSO network consists of a distributed set of peers, the computational effort needed to compare the cases is spread among the peers, eliminating a single computational bottleneck of traditional centralized CBR systems and allowing parallelization of the case retrieval.

The proposed approach was evaluated using five case-bases of real life E-Commerce advertisements from various application domains. We compared the proposed approximated similarity-based retrieval algorithm with the traditional exhaustive retrieval algorithm. The results showed that the former significantly reduces the number of cases compared at the retrieval stage, while preserving the essential quality [in terms of precision and recall Salton and McGill (1983)] of the retrieved cases. Hence, the contributions of this work are two-fold. First, we propose a novel notion of pure decentralized storage of the case-base in a P2P environment. Second, we develop and evaluate an efficient approximated algorithm for retrieval of the most similar cases over the above decentralized storage of the case-base.

The rest of this paper is structured as follows. In Sects. 2–5 we review the related research: Sect. 2 presents semantic approaches to P2P data management, Sect. 3 describe the unspecified P2P data management using UNSO, Sect. 4 describes the retrieval of nearest neighbors in P2P networks, and Sect. 5 discusses speed-up and distributed CBR techniques. In Sects. 6–8 we present the original contribution of our work extending the line of research presented in Sects. 2–5: Sect. 6 presents the representation and storage of cases using UNSO, Sect. 7 discusses the proposed retrieval algorithm, and Sect. 8 presents and analyzes the experimental evaluation. Finally, Sect. 9 we concludes the paper.

2 Semantic data management in peer-to-peer systems

The first P2P systems were designed for large-scale data sharing (Androutsellis-Theotokis and Spinellis 2004). Applications, such as Napster Inc, Freenet Clarke et al. (2000) and Gnutella Adar and Huberman (2000), allowed users to download multimedia data uploaded by other users. Performance of these systems suffered from severe scalability problems. In Napster, a cluster of central servers maintained the indices of the data uploaded by the users. Flooding search algorithm of Gnutella did not allow communication-efficient retrieval of the data over a heterogeneous set of users. Freenet, despite being fully decentralized and employing efficient routing algorithms, could not guarantee reliable and unambiguous data management. This has led to the development of content-addressable P2P systems (Milojicic et al. 2002).

A number of content-addressable P2P systems for data sharing, such as CAN (Ratnasamy et al. 2001) and Pastry (Rowstron and Druschel 2001) were developed. These applications address the scalability issues and implement a self-organizing infrastructure for fault-tolerant routing using a Distributed Hashing Table (DHT) (Harren et al. 2002). In DHT-based systems, every user and data object is assigned a unique identifier called *user* and *key*, respectively. Since the data objects are provided and stored by the users, every user can virtually be represented by a set of the provided data objects, i.e., $user_i = \{key_{i1},\ key_{i2}, \ldots, key_{im}\}$.

The data objects are inserted into the system by mapping their keys to the nodes of the overlay P2P network. This is achieved using a globally-known $put(key_{ij})$ operation, which assigns $user_i$ invoking it as the provider of the data object identified by key_{ij}. This assignment is done by coupling the user identifier $user_i$ with the network node $node_k$, which was obtained by applying the globally-known hashing mechanism on key_{ij}. That is, globally-known hashing mechanism converts the data object identifier key_{ij} into the node $node_k$ that is coupled with the identifier of the user providing the data object. Other users can discover a data object identified by key_{ij} through $get(key_{ij})$ operation applying the same hashing mechanism as $put()$ operation. In this case, the hashing mechanism returns the node $node_k$.

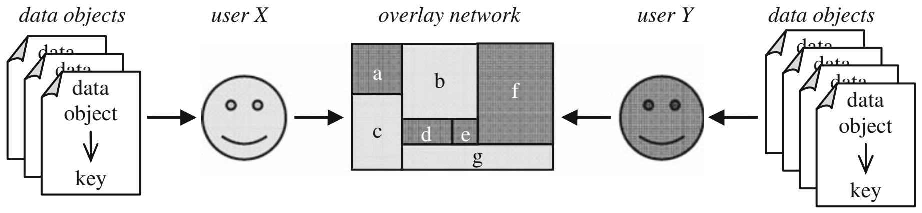


Fig. 1 Management of data objects in DHT-based P2P middleware

Hence, DHT-based systems facilitate decentralized management and search of data objects, where the unique description of the data object key_{ij} and the hashing mechanism of $put()$ and $get()$ operations guarantee proper functionality.

For example, consider two users, X and Y, providing seven data objects (three provided by X and four provided by Y in a CAN-based system Ratnasamy et al. 2001) exploiting two-dimensional overlay network[1] (see Fig. 1). The keys of the data objects are mapped to the nodes of the middleware using the hashing mechanism: the data objects provided by X are mapped to nodes b, c, and g, whereas the data objects provided by Y to nodes a, d, e, and f (shown in different tones). When other users search for one of these data objects, the hashing mechanism points them to the appropriate node of the overlay network, and the node—to the user providing the data object.

In comparison to other P2P systems DHT-based systems are highly scalable, and provide a robust, self-organizable, and pure decentralized structure. Due to the effective Plaxton mesh routing algorithm (Plaxton et al. 1997), which routes the messages to the relevant peers only and does not use an expensive network flooding, their overall traffic is considerably low (Rowstron and Druschel 2001). However, DHT-based systems basically rely on hashing-based $put()$ and $get()$ operations and this results in two limitations:

- Exact-matching lookup. Keys of two similar but not identical data objects, key_1 and key_2, are considered completely different. Hence, only the searches specifying the exact *key* used while inserting a data object will discover it. This limitation hampers the data management characteristics of DHT-based systems, and leads to an uncontrolled redundancy in the data object repository.
- Single key lookup. The above $put()$ and $get()$ operations handle a single *key* only, such that the data object are described by a single string. Although the *key* might be represented as a concatenation of the substrings of its parts, changing one of the substrings will prevent discovering it, as the hashing mechanism will assign it to a different node.

These limitations have lead to the development of a more complex P2P networks, built upon peers using their own schemata to describe the data objects. This approach is referred to as semantic *ontological* data management, where the ontology constitutes the unified view of the schemata used by the users. In general, ontology is one of the key concepts in semantic data management, defined as "a formal shared conceptualization of a particular domain of interest" (Gruber 1993). It acts as a standardized reference model providing human-understandable and machine-processable semantic mechanisms, which allow various systems to collaborate efficiently. Two techniques for ontology-based data management in P2P networks were proposed in HyperCup (Schlosser et al. 2002) and UNSO (Ben-Asher and Berkovsky 2006).

[1] For the sake of clarity Fig. 1 shows the zones rather than the nodes of the CAN network.

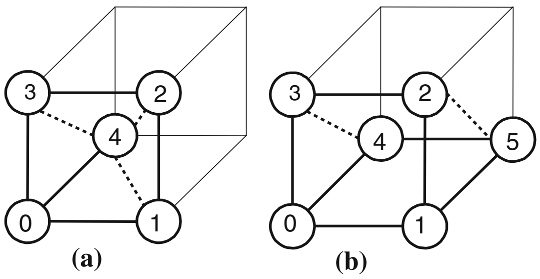


Fig. 2 **a** Complete hypercube topology. **b** Insertion of a new data object

HyperCup (Schlosser et al. 2002) proposes a flexible ontology-based pure decentralized P2P system generating a hypercube-like graph of nodes. To manage the data objects, HyperCup exploits predefined domain ontology, such that the dimensions of the hypercube match the concepts of the ontology, i.e., domain features. As a result, every data object described using the ontology is mapped to a single node of the hypercube. This mapping couples the node of the hypercube with the user providing the data object. The data objects can be discovered by other users using the same ontology. Note that the ontology-based mapping of HyperCuP inherently determines the order of values of a feature, e.g., $small = 1$, $medium = 2$, and $large = 3$ for the feature *size*. Therefore, HyperCup facilitates accurate similarity metrics for approximated retrieval and ranking of data objects.

HyperCup nodes are interconnected, such that every hypercube node stores lists of its immediate neighbors in various dimensions. For example, consider a node (x, y, z) of the 3-dimensional hypercube. It is connected to six neighbors, i.e., nodes $(x + 1, y, z)$, $(x - 1, y, z)$, $(x, y + 1, z)$, $(x, y - 1, z)$, $(x, y, z + 1)$, and $(x, y, z - 1)$. Note that the $+1$ and -1 in the node addresses do not mean any numeric order or semantic similarity, but a logical neighborhood relation only. In terms of the data objects, consider the following simple flat ontology of cars having three features: *manufacturer, color*, and *produced*. According to it, the node storing the data object <$manufacturer$:BMW, $color$:*black*, $produced$:1987> is connected to nodes storing black cars produced by various manufacturers (but not by *BMW*) in 1987, *BMW* cars of various colors (but not black) produced in 1987, and black cars produced by *BMW* in various years (but not 1987).[2]

HyperCuP also proposes a dynamic algorithm for construction and maintenance of the hypercube. It is based on the idea that a user providing a particular data object can be coupled not only the node to which the provided data object is mapped to, but also a set of *virtual* neighbor nodes, which have not yet been coupled with other users and data objects. This is required to simulate the missing nodes and maintain connectivity. For example, consider a 3-dimensional hypercube with five data objects (see Fig. 2a). In this case, node 4 simulates three missing nodes of the hypercube. The simulated nodes are schematically illustrated by the dashed edges 1–4, 2–4 and 3–4. When a new data object is inserted and assigned to one of the missing nodes, the new node starts functioning as a real hypercube node. For example, consider a new data object inserted to node 5 (see Fig. 2b). It becomes the neighbor of nodes 1 and 4, and simulates the missing neighbor of node 2. The detailed presentation of the hypercube maintenance and examples of such insertions can be found in Schlosser et al. (2002).

As we already mentioned, HyperCuP ontology is fixed and defined prior to the insertion of the data object. This requires the application domains to be modeled by human domain

[2] Connectivity of the overlay nodes is guaranteed by the connectivity of the underlying P2P network (Schlosser et al. 2002) and is out of scope of this work.

experts and does not support a setting, where various data objects are dynamically inserted by the users. Also, it inherently implies that the ontology is managed centrally and cannot be modified over time. These characteristics of HyperCuP contradict the decentralized and highly dynamic nature of P2P networks and call for a more flexible approach to the representation of the data objects.

3 Unspecified data management

UNSO (UNSpecified Ontology) (Ben-Asher and Berkovsky 2006) extends the above ideas of semantic P2P data management using a hypercube-like overlay network graph. The main contribution of UNSO is that it addresses and resolves the above limitation of HyperCuP, i.e., the usage of predefined and fixed domain ontology. Unlike HyperCuP, UNSO assumes that the domain ontology is not fully defined before objects are inserted and parts of it can be incrementally defined when inserting the data objects. Hence, the description of a data object is relatively flexible and is modeled as an *unspecified* vector. Unspecified means that the data objects are described by a list of features and their values: <$feature_1$:$value_1$, $feature_2$: $value_2, \ldots,$ $feature_n$:$value_n$>, where $feature_i$ corresponds to feature i in the description of the data object and $value_i$ is its respective value.

Furthermore, UNSO generalizes the ontological data management of HyperCuP. Since the list of features and values is not fixed and can grow incrementally, two hashing functions are used to map the data objects to their nodes of the hypercube. The first function $hash_1$ maps $feature_i$ to a certain dimension of the hypercube, while the second $hash_2$ maps $value_i$ to a numeric coordinate within this dimension. For example, consider the following unspecified description: <*manufacturer:BMW, color:black, produced:1987*>. It is inserted by applying $hash_1$ $(manufacturer)$, $hash_1$ $(color)$, and $hash_1$ $(produced)$ to obtain the dimensions of the hypercube, while $hash_2(BMW)$, $hash_2$ $(black)$ and $hash_2(1987)$ determine the numeric coordinates within these dimensions. Thus, the coordinate of the data object in the dimension $hash_1$ $(manufacturer)$ is $hash_2(BMW)$, in the dimension $hash_1$ $(color)$–$hash_2$ $(black)$, and in the dimension $hash_1$ $(produced)$–$hash_2(1987)$. Hence, UNSO can manage objects, *features* and *values* of which were not anticipated by the domain ontology. In addition, hashing-based mapping of UNSO allows the users to insert data objects with no particular order of features, further increasing flexibility.

On the other hand, this introduces a new issue, as the users may use different terms to describe the same semantic concept, e.g., synonyms, hyponyms, or hypernyms. To address this issue, the features and values are standardized using WordNet (Fellbaum 1998). In WordNet, English nouns, verbs, adjectives and adverbs are organized into synonym sets, each representing one lexical concept. To minimize ambiguity, the terms that appear in the descriptions of the data objects are substituted by their most frequent synonyms. For example, the terms *auto*, *automobile*, *motorcar*, and *vehicle* are substituted by their most frequent synonym *car*. Thus, similar but not identical terms are replaced by a single representative term.

Many features of the data objects are not independent and appear only if another feature, specific value, or even a certain combination of features and values appears in the description of the data object. For example, consider a car's *ABS* (Anti-lock Braking System) feature. Since only the cars produced after 1978 may be equipped with an ABS, this feature cannot appear in the descriptions of cars produced before 1978. To handle these dependencies in a flexible manner, the data objects are organized as a *hierarchical* multi-layered rather than a *flat* structure. This converts the hypercube structure of HyperCup into

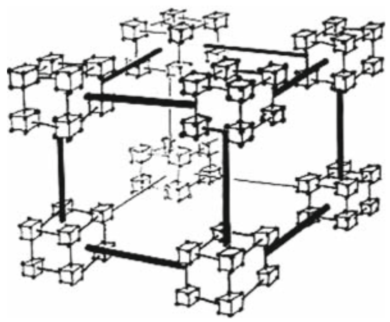


Fig. 3 Multi-layered hypercube (MLH) of UNSO

a hypercube-like structure, whose nodes recursively consist of other hypercubes. This structure is referred to in Ben-Asher and Berkovsky (2006) as a multi-layered hypercube (MLH). Figure 3 schematically illustrates the organization of the MLH and shows its recursive structure, according to which the nodes of the higher-level hypercubes consist of the lower-level hypercubes.

If the number of features in the description of a data object is smaller than the number of MLH dimensions, then the description is expanded by assigning an *unknown* value denoted by ∅ to the missing features. For example, consider a data object <*manufacturer:BMW, color:black* > inserted into the above three-dimensional hypercube. It is first expanded to <*manufacturer:BMW, color:black, produced:*∅> and then inserted into the MLH. Conversely, if a new data object introduces a new feature that has not appeared yet in the existing data objects, a new dimension is added to the lowest-level hypercube of the MLH. Note that the new dimension is added only to a single lowest-level hypercube, whose location reflects the other features mentioned in the data object that introduces a new feature, whereas the rest of the MLH is not affected. This constitutes one of the advantages of UNSO over HyperCup, as the structure of the MLH is highly dynamic and reflects the features that appear in the data objects. For example, consider a three-dimensional MLH, whose dimensions are *manufacturer, color, produced*. Consider two of its low-level hypercubes: one stores data objects representing cars and the other—mobile phones. The dimensions of the low-level hypercubes may represent features *engine volume, gear*, and *ABS* for cars, and *network, frequency*, and *display* for mobile phones. Furthermore, the overall number of features in the low-level MLHs may be 20 for cars and only 10 for mobile phones.

The differences between a fixed ontology and UNSO are summarized in Fig. 4. The upper part shows the mapping of a data object described according to a predefined ontology of HyperCuP to a hypercube node. The ontology has three features $<f_1, f_2, f_3>$ having predefined sets of values: $val_{11}, val_{12}, \ldots, val_{1n}$ for feature f_1, $val_{21}, val_{22} \ldots, val_{2n}$ for f_2, and $val_{31}, val_{32}, \ldots, val_{3n}$ for f_3. The features are assigned to the dimensions of the hypercube using the $hash_1$ function, while their respective values are assigned numeric coordinates within these dimensions using the $hash_2$ function. Hence, the data object $<f_1 : val_{1i}, f_2 : val_{2j}, f_3 : value_{3k}>$ is mapped to the location denoted by $<val_{1i}, val_{2j}, val_{3k}>$. Conversely, in UNSO (the bottom part) the description of a data object is partitioned into several levels of the MLH. For example, consider the data object $<f_{11} : v_{11}, f_{12} : v_{12}, f_{13} : v_{13}> + <f_{21} : v_{21}, f_{22} : v_{22}, f_{23} : v_{23}> + <f_{31} : v_{31}, f_{32} : v_{32}, f_{33} : v_{33}>$, which has already been partitioned into three levels of the MLH. To insert this data object into the MLH, the features are hashed into dimensions of the hypercube, the values are hashed into

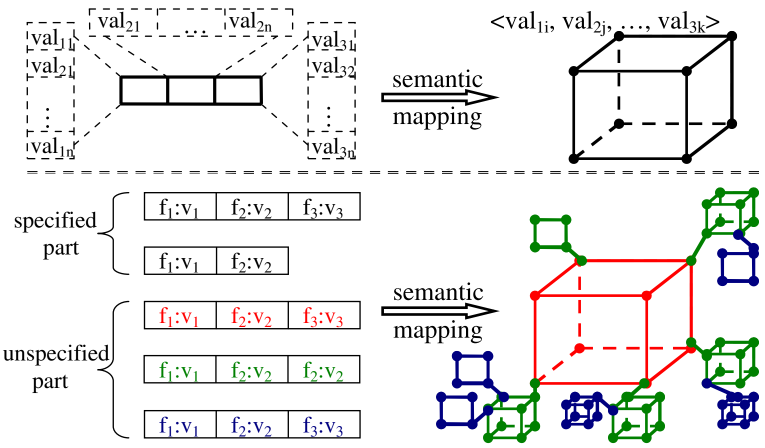


Fig. 4 Generalization of the fixed ontology of HyperCuP into UNSO

numeric coordinates within these dimensions, and the data object is inserted, for instance, into the node denoted by X.[3]

In summary, UNSO provides a hypercube-based platform for a fully decentralized management of heterogeneous data objects. UNSO allows the data objects to be described in a flexible manner, not implying any centralized ontology. New features and values can be introduced: new features are converted into new dimensions of the low-level hypercubes, while new values are just mapped to their coordinates in these dimensions. The MLH structure allows a flexible description of the data objects, expressing dependencies between various features and values. It also exhibits a denser distribution of data objects than the flat hypercube, since the dimensionality of the MLH is lower, and as such, is easier to maintain in a P2P environment. Hashing-based mapping mechanism of UNSO allows the order of features and values in the data objects to be disregarded. Semantic standardization using WordNet minimizes heterogeneity and facilitates accurate data management (Ben-Asher and Berkovsky 2006).

As a negative effect of hashing in UNSO, we would like to mention that any order and similarity relations among the values of a feature are lost. Despite this, UNSO supports the notion of concept clustering (Schlosser et al. 2002), as data objects that are identical with respect to a subset of features are mapped to locations having a subset of identical coordinates and values. For example, data objects <*manufacturer:BMW, color:black, produced:1987*> and <*manufacturer:BMW, color:white, produed:1987*> will be mapped to nearby locations, as their *manufacturer* and *produced* coordinates will identical, and *color* coordinates will differ. Also data objects <*manufacturer:BMW, color:black, produced:1987*> and <*manufacturer:BMW, color:black, accidents:0*> will be mapped nearer than two arbitrary data objects, as two out of three features overlap.[4]

[3] For the sake of clarity, low-level hypercubes are shown as separate hypercubes attached to the nodes of the higher-level hypercubes. In fact, the structure of the MLH is recursive, i.e., the nodes of high-level hypercubes consist of the low-level hypercubes.

[4] Clustering may not be accurate due to hashing collisions, e.g., if $hash_2(1987) = hash_2(1978)$, different data objects will be mapped to the same location. Probability of the collisions decreases with the number of features and the range of the hashing function.

4 Retrieval of nearest neighbors in P2P networks

The issue of similarity-based retrieval of nearest neighbors in decentralized P2P environments was studied in a number of works. FuzzyPeer, a P2P system facilitating execution of similarity-based queries through a broadcast-based flooding of messages, is discussed in Kalnis et al. (2006). The system supports a wide variety of queries, such as range, approximated, multi-attribute, and K-Nearest Neighbors queries. To optimize the expensive broadcast mechanism, FuzzyPeer proposes a frozen broadcast of messages, where the messages are conditionally routed within the system basing on a set of previous messages routed through the nodes. However, the communicational overhead of FuzzyPeer is relatively high, and multi-attribute queries are limited to a set of a-priori defined attributes only.

From the communicational overheads perspective, P2P systems facilitate more efficient execution of approximated and range queries over multi-dimensional data. This approach is based on an indexing scheme, called Locality-Sensitive Hashing (LSH) (Bawa et al. 2005). The basic idea behind LSH is as follows: the data objects are hashed by special locality-sensitive hash functions, such that the probability of similar objects to be hashed to the same bucket is higher that the probability of dissimilar objects. When a query is issued, it is also hashed and the data objects which were hashed to the same bucket are the best candidate answers. Their real similarity to the query is computed, and the most similar are returned as answers to the query. Although LSH and an LSH-based prefix tree called LSH-forest provide an efficient middleware for multi-attribute queries in a P2P environment, its main limitation is that the set of attributes used both in queries and data object descriptions is fixed and predefined.

A number of works focus on issuing queries in hierarchically structured semi-centralized P2P networks (Milojicic et al. 2002). In these networks the peers are partitioned to several layers, whereas the upper-layer peers, i.e., the super-peers, serve as the mediators for the lower-layer peers and handle parts of their functionalities. For example, in Tran (2005) the authors propose EZSearch, a pure distributed P2P search mechanism with a low search overhead supporting both range and K-Nearest Neighbors queries. This is achieved by using a hierarchical locality-preserving index, where the nodes are grouped to clusters, each partially managing the underlying nodes. To support the search operations, the queries are forwarded in the hierarchy to the proper cluster and then to the nodes storing the relevant data objects. However, the semi-centralized structure of EZSearch hampers the scalability of the system.

Specific hierarchy-based application of K-Nearest Neighbors retrieval in P2P networks is discussed in Pouwelse et al. (2005). This work presents a distributed P2P approach for recommending TV programs by exploiting the play-lists stored by the other users in the network. According to it, the recommendations are built by combining the play-lists of like-minded users, while the similarity of users is determining by comparing their play-lists. However, also in this work the peers are organized in a hierarchical semi-centralized structure limiting the scalability of the system. Another domain-specific application is discussed in Tanin et al. (2005), which describes an application aimed at answering approximated spatial queries. This paper proposes an efficient algorithm handling approximated queries for K-Nearest spatial objects, where the similarity is measured by the geographic distance. Although the proposed algorithm exploits a distributed spatial index that does not rely on a central server, the index is fixed and defined a-priori by the system designers.

5 Speed-up techniques and distributed case-based reasoning

The retrieval of cases that are most similar to the case to be solved is one of the main stages in a CBR problem solver (Aamodt and Plaza 1994). It is referred to in the literature as similarity-based retrieval (de Mantaras et al. 2005). A basic requirement for the similarity-based retrieval is the definition of an appropriate similarity function, which allows to compute the similarity of two cases in a case-base, and then to identify the most useful similar cases. In many CBR applications (and we followed this approach in our experiments), the similarity is computed through so-called *surface* features, which are provided as part of the case description and are typically represented as feature vectors of attribute-value pairs (de Mantaras et al. 2005). Retrieval of cases based on a flat representation of surface features is referred to as *surface* retrieval. However, this process has complexity $O(nm)$, where n is the number of cases and m is the average length of the vector, which may be an unacceptable overhead if n is large. Hence, many prior works developed approaches that reduce the computational cost of finding similar cases or spread the computational overhead over a number of loosely coupled units.

One approach for reducing the retrieval time relies on a smart organization of cases. For example, Wess et al. (1993) proposes a sound and complete approach to the similarity-based retrieval, in which the organization of cases is based on their similarity. K-d trees (Friedman et al. 1977) are used to split the case-base into groups of cases, such that each group consists of cases that are similar to each other according to a given similarity metric. To improve the accuracy of the retrieval and to guarantee that the most similar cases are retrieved, the retrieval algorithm computes similarity bounds to determine which groups of cases should be considered first. These approaches work in vector spaces in which the features are assigned real values and cannot be generalized to generic similarity metrics for both numeric and symbolic attributes. In the case study illustrated later we rely on the mapping of the original representation space onto the MLH. In the MLH, expanding chain of neighborhood sets with an increasing number of modified coordinates provides an index for the search of cases similar to the target case. Hence the role of this chain is similar to that of the static K-d tree partition of the search space.

The ideas of distributed problem solving in CBR arose in multi-agent CBR systems, such as those proposed in Nagendra et al. (1996) and Plaza et al. (1996). In Nagendra et al. (1996), the authors present a formal model for cooperative retrieval and composition of cases, in which sub-cases are distributed across agents in a multi-agent system. The retrieval process involves iterative negotiation between the agents, aimed at resolving conflicts, relaxing as few soft and satisfying as many hard constraints as possible. The experimental results show that the retrieval time of the negotiating algorithm is linear with the case-base size, whereas the difference between the negotiating and non-negotiating retrieval times increases with it. We observe that in our approach we do not partition a case (request) between multiple agents, but rather search in the nearby nodes for other cases that differ in a subset of features.

In Plaza et al. (1996), the authors present two modes of cooperation between the agents: *distributed* CBR (DistCBR) and *collective* CBR (ColCBR). Both of them are based on solving a problem using knowledge learned by other agents. The difference between the two lays in the solution method that is used: the method of the remote agent in the former, whereas the method of the agent solving the problem in the latter. Thus, in DistCBR the agent that needs to solve a problem, distributes it to the other agents, and each agent resolves the problem using its own methods in a fully distributed manner. Conversely, in ColCBR the agent that needs to solve a problem distributes the solution method jointly with the problem to be solved to other agents. In our approach, a similarity-based query to the P2P infrastructure is managed by a node that receives the query and then a search for relevant cases is spread in the network to discover similar cases located near the target case.

A new approach for a personalized distributed CBR, called *collaborative* CBR, or in short CCBR, was proposed in McGinty and Smyth (2001). This approach is based on the idea of Collaborative Filtering, which assumes that the users that agreed in the past will also agree in the future (Herlocker et al. 1999). Hence, CCBR implements a travel-planning multi-agent CBR system that exploits the knowledge of highly similar (in terms of past traveling) agents. Applicability of the proposed CCBR approach was experimentally evaluated and showed that the quality of personalized traveling plans outperforms the quality of traditional non-personalized ones. This approach can be considered as an example of how the P2P infrastructure we propose can be exploited for matching the similarity of *problem solvers* at different network locations, rather than matching the *problem definition* with solution cases located in other nodes of the network.

Another approach, called *multi* CBR, is presented in Leake and Sooriamurthi (2001) and Leake and Sooriamurthi (2003). It refers not only to applying multiple case-bases to resolve a problem, but also to determining the conditions in which knowledge from the other case-bases is beneficial. In Leake and Sooriamurthi (2001), the authors motivate distributed CBR by the problems inherent in merging multiple case-bases into a single one: availability, efficiency, standardization, and maintenance, which apparently are the advantages of a distributed storage. Experimental evaluation analyzed a few policies for combining knowledge of multiple case-bases and showed that the ability to dispatch cases to a remote case-base improves system performance, given that the knowledge transfer is supported by cross case-base adaptation. In Leake and Sooriamurthi (2003), the authors further investigate the reasons for keeping multiple case-bases even if merging them is feasible. They showed that it is not beneficial to merge, because multi CBR can exploit the tradeoffs between the similarity of problems and similarity of solution contexts. We believe that such an approach can be considered as an extension of the heuristics proposed in this paper, i.e., that cases more useful for the problem solving activity are located nearby in terms of network topology.

In Plaza and McGinty (2005), the authors propose a classification of distributed CBR by considering two dimensions describing (1) how the knowledge is organized within the system and (2) how it is processed. For each dimension they provide two values: single versus multiple case-bases for knowledge organization and single versus multiple agents for the knowledge processing. The approaches described in McGinty and Smyth (2001), Nagendra et al. (1996) and Plaza et al. (1996) are examples of the most complex type, i.e., multi agents and multi case-bases. Examples of the single agent, multi case-bases are popular in CBR. Cases can be partitioned into case-bases storing cases at different level of abstraction, as in Branting and Aha (1995) and Smyth and Cunningham (1996), or by competence, as in Leake and Sooriamurthi (2001). In multi-agents and single case-base the case storage is centralized, but a collection of agents cooperates to solve a problem, implementing the standard CBR problem solving cycle in a distributed manner. XML description of cases is used for the communication of agents in client-server architecture (Watson and Gardingen 1999; Coyle et al. 2004).

We would like to stress that a P2P approach for storing and retrieving cases actually refers to both dimensions mentioned in Plaza and McGinty (2005). The cases are physically provided and stored by the users, and they can be accessed and retrieved by allocating the cases to the nodes of the overlay network. Furthermore, retrieval of the most similar cases can be performed using the basic look-up functionality of the P2P infrastructure (will be shown later).

6 Representation and similarity of cases in UNSO

In the rest of this paper we will present our original contribution to the veins of research reviewed in the previous sections. We first describe UNSO-based representation and similarity of CBR cases, then present an algorithm for a similarity-based retrieval of cases, and finally present the evaluation of the proposed algorithm.

In this work, we use the unspecified description of the data objects as a dynamic list of features and their values as case representation language in a CBR system. In pure decentralized environments lacking a centralized management or ontology, different users might represent similar cases in different ways and/or by using different terms representing the same semantic concepts. Nevertheless, UNSO implicitly groups similar cases and stores them in nearby locations with respect to the overlay network topology. Such representation of cases and the fact the similar cases are grouped allows similarity-based retrieval of cases to be preformed as a localized search starting at the location of the target case in the overlay network (the details of the retrieval process will be elaborately discussed later on).

In this work we consider a case-base that stores E-Commerce advertisements (or ads, in short) from different application domains. The ads are divided into two categories: *supply* ads where users offer products or services in exchange for payment, and *demand* ads where users look for products or services offered by others. One of the main goals of a typical E-Commerce system is to match the appropriate demand and supply ads. In a decentralized setting, having no predefined format for inserting the ads, the matching is harder to achieve, as the system should find matching ads basing on partial, incomplete, or incompatible descriptions of the ads. In the context of a CBR application, demand ads represent the problems to be solved, while supply ads represent the possible solutions. The system should support the retrieval of the most-similar supply ads for each demand (target) ad.

Efficient and accurate retrieval of the most-similar cases is one of the primary goals of a CBR system, as it facilitates further reuse and adaptation stages (Watson 1997). Two basic policies for case retrieval are typically exploited:

- Retrieve the whole set of cases, whose similarity with the target case is above a given threshold β—typically performed by computing the similarity degree of each target case and returning the case if it is above β.
- Retrieve a set of K cases most-similar to the target case—typically performed by computing the similarity degree of each target case, ranking the cases according to their similarity, and returning K cases at the top of the list.

Essentially, these two policies are similar, as they retrieve the most similar cases by exhaustively comparing the target case with all the cases in the case-base, and in many conditions these techniques are interchangeable. However, in certain conditions, one of them may be preferred over the other. For example, in a case-base where the number of similar cases is high, retrieving all the cases whose similarity is above β may result in too many cases, and limiting this size of the retrieved set may be preferred. On the contrary, in a smaller case-base, the retrieval of K most-similar cases may yield an insufficient number of similar cases, and retrieving all the cases whose similarity is above a certain threshold may be preferred.

The above policies require a stable similarity metric between the cases to be defined (Bernstein et al. 2005). In this paper, the similarity of cases c_1 and c_2 is computed by $(1 - dist(c_1, c_2))$, where $dist(c_1, c_2)$ is a normalized [between (0) and (1)] distance metric. When cases are represented by a list of $feature_i$:$value_i$ pairs, the distance between the cases is computed as a weighted combination of the distances between individual features

$$dist(c_1, c_2) = \left[\sum_{i=1}^{|F|} w_i \cdot dist(c_1^i, c_2^i)^2 \right]^{1/2}.$$

F denotes the union of the features that appear in the cases, w_i is the weight of a feature f_i, and $dist(c_1^i, c_2^i)$ is the normalized [between (0) and (1)] local distance metric for the values of feature f_i.

To compute the local distance between two values we consider the type of the feature (Coyle et al. 2004). For Boolean features, the distance is a trivial comparison between two values giving (0) if the values are equal or (1) otherwise. For the free-text features, the distance is computed as the difference between the numeric representations of the values (reflecting their similarities or order relation, e.g., assigning $white = 0$, $gray = 0.5$ and $black = 1$ for the *color* feature) or using a matrix assigning a distance value to each pair of values. Although in some domains these assignments can be performed manually by a human expert, there are domains in which it is unclear whether this will be feasible and whether the produced distance metric will be reliable. Hence, following other CBR approaches, such as in Richter (1992) and Wilson and Martinez (1997), we define the distance metric for the free-text features similarly to the metric of Boolean features:

$$dist_{Boolean}\left(c_1^i, c_2^i\right) = dist_{Free-Text}\left(c_1^i, c_2^i\right) = \begin{cases} 0 & c_1^i = c_2^i \\ 1 & \text{otherwise} \end{cases}$$

This rather harsh metric requires exact matching of the values. It can be substituted by more lenient metrics, e.g., assigning an average (local) distance between cases having non-matching values for a feature (Bogaerts and Leake 2004). Although such local metrics may affect the accuracy of the retrieval, experimenting with them falls beyond the scope of this work. For the numeric features, the distance between the values is computed as the absolute value of the difference normalized by the maximal range of values R of a feature:

$$dist_{Numeric}(c_1^i, c_2^i) = \frac{|c_1^i - c_2^i|}{R}$$

Analyzing the possible types of features in the unspecified descriptions of the ads showed that most of the features are either numeric, free-text, or Boolean. Thus, in this work we exploit only these local similarity metrics, and do not define metrics for more complex types of features (Coyle et al. 2004).

Since users may describe the same case using different features, even after the standardization the cases may differ in terms of the mentioned features. Hence, the similarity metric between cases should be able to compute similarity of cases having only partially overlapping features. For this, we define the distance between the values of a feature to be 1 if the feature appears in one case, but does not appear in the other:

$$dist'\left(c_1^i, c_2^i\right) = \begin{cases} dist\left(c_1^i, c_2^i\right) & \text{feature } f_i \text{ appears in both cases} \\ 1 & \text{otherwise} \end{cases}$$

Consider the following case similarity computation. Two cases c_1 and c_2 are cars ads: $c_1 =< manufacturer: Ford,\ model: Focus,\ doors: 4,\ color: white,\ accidents: 0,\ price: 15000,\ ABS: true >$ and $c_2 =< manufacturer: Ford,\ doors: 2,\ model: Mustang,\ pro-duced: 2003,\ price: 19000,\ ABS: true >$. The overlapping features for these cases are *manufacturer, model, doors, ABS*, and *price*. The features *doors* and *price* are numeric, *manufacturer* and *model* are free-text, and *ABS* is Boolean. Let us assume that the range of *doors*

Table 1 Distance between the features of c_1 and c_2

Feature	Manufacturer	Model	Doors	Color	Accidents	Price	ABS	Produced
c_1	Ford	Focus	4	White	0	15000	True	unknown
c_2	Ford	Mustang	2	unknown	unknown	19000	True	2003
$\mathrm{dist}'(c_1^i, c_2^i)$	0	1	0.67	1	1	0.04	0	1

is 2–5 and the range of *price* is 0–100,000. Table 1 summarizes the distances between the features of these cases calculated as follows:

Hence, the distance between these two cases is calculated as follows:

$$dist(c_1, c_2) = \Bigg[(w_{manufacturer} \cdot 0)^2 + (w_{model} \cdot 1)^2 + \left(w_{doors} \cdot \frac{4-2}{5-2}\right)^2 + (w_{color} \cdot 1)^2 + (w_{accidents} \cdot 1)^2 + \left(w_{price} \cdot \frac{19000-15000}{100000-0}\right)^2 + (w_{ABS} \cdot 0)^2 + (w_{year} \cdot 1)^2\Bigg]^{1/2}$$

7 Similarity-based retrieval algorithm

In this work we propose exploiting the dynamic infrastructure of UNSO and grouping of similar cases as an indexing approach. Intuitively, we base the proposed retrieval algorithm on the assumption that similar cases are located in nearby nodes. Hence, retrieval of similar cases is performed as a localized search in the MLH, starting from the node in which the target case to be solved is located and iteratively checking neighbor nodes and comparing the cases stored there with the target case. Figure 5 presents the pseudo-code of the algorithm for the retrieval of cases whose similarity with the case to be solved is above a given threshold β.

Initially, the algorithm determines the location of the *TARGET-CASE* to be solved using the hashing-based mechanism for inserting cases into the MLH (steps 1–2 of the pseudo-code). Then the algorithm analyzes the candidate *TEST-CASES* stored in the neighbor nodes of the overlay network by assessing the similarity between each *TEST-CASE* and the *TARGET-CASE*

```
Retrieve (TARGET-CASE, $\beta$)
(1)  map TARGET-CASE to the hypercube of dimension $n$
(2)  assume the location of TARGET-CASE is $(h_1,\dots,h_n)$
(3)  let RETRIEVED-CASES be a set of cases, initially empty
(4)  for each dimension $i$ from 1 to $n$
(5)    for each $x$ in the range of values for coordinate $i$
(6)      let CURRENT be the set of cases stored in
         the location $(h_1,\dots,h_{i-1},x,h_{i+1},\dots,h_n)$
(7)      for each TEST-CASE $\in$ CURRENT
(8)        if similarity(TARGET-CASE, TEST-CASE) > $\beta$
(9)          RETRIEVED-CASES = RETRIEVED-CASES $\cup$ {TEST-CASE}
(10) return RETRIEVED-CASES
```

Fig. 5 Algorithm for retrieving cases with similarity metric above β

(steps 4–7 of the pseudo-code). The similarity between each one of the *TEST-CASES* and the *TARGET-CASE* to be solved is calculated using the above distance metrics. If the similarity value between the *TARGET-CASE* and the *TEST-CASE* is higher than the given threshold β, the *TEST-CASE* is added to the set of cases with the required similarity *RETRIEVED-CASES* (steps 8–9 of the pseudo-code). Finally, the whole set of appropriate cases *RETRIEVED-CASES* is returned (step 10 of the pseudo-code).

Note that the retrieval process is conducted in the MLH by iteratively propagating the queries from the target node to its neighbor nodes and so on. That is, consider a target case mapped to the location (x, y, z) of a 3-dimensional hypercube. This node is connected to 6 immediate neighbor nodes denoted by $(x + 1, y, z)$, $(x - 1, y, z)$, $(x, y + 1, z)$, $(x, y - 1, z)$, $(x, y, z + 1)$ and $(x, y, z - 1)$. Consider the search over the first dimension, i.e., in the nodes $(x + 1, y, z)$ and $(x - 1, y, z)$. When these nodes receive the similarity computation request, the cases stored there are checked and the request is propagated to their neighbor nodes, i.e., to $(x + 2, y, z)$ and $(x - 2, y, z)$. The same propagation occurs also in the other dimensions of the 3-dimensional hypercube. Due to the distributed structure of the overlay network, when the neighbor nodes receive the similarity computation request, all the computations (and also the propagation of the request to their neighbor nodes) are performed in parallel. Thus, the retrieval can be described as an expanding search, where the similarity computations of the cases and the communication activities required by the retrieval are parallelized and distributed between the MLH nodes.

For example, consider the following target case $c_t =< manufacturer{:}BMW, color{:} black, doors{:}2 >$ and a 3-dimensional MLH storing the case-base. Assume that c_t is mapped to the MLH node (5, 6, 7) The proposed algorithm will compare c_t with the cases stored in the nodes (*, 6, 7), (5, *, 7), and (5, 6, *), where * denotes any possible value. To compare c_t with the cases stored in (*, 6, 7) nodes, the node (5, 6, 7) propagates the similarity computation request to nodes (4, 6, 7) and (6, 6, 7). Upon receiving the query, these nodes (i) forward the request to their neighbor nodes (3, 6, 7) and (7, 6, 7), (ii) compute the similarity between the target case c_t and the cases stored in their nodes, and (iii) back propagate the similarity value to the node from which the query was received. Such propagation of the similarity computation request occurs also over the other dimensions.

The retrieval of K most-similar cases is performed similarly. However, the size of the *RETRIEVED-CASES* set is limited to K. Since at any given point of time *RETRIEVED-CASES* stores K most-similar candidates for being the most similar cases, these cases are sorted according to their similarity with the *TARGET-CASE*. Hence, every *TEST-CASE* is added to *RETRIEVED-CASES* in a way that keeps the *RETRIEVED-CASES* set sorted. Figure 6 presents the pseudo-code of the algorithm for retrieval of K cases most similar to the case to be solved.

Let us denote by Δ the maximal number of coordinates, whose values are modified with respect to the coordinates of the node, to which that the target case would be mapped.[5] The above algorithms retrieve the most similar cases, where the value of up to $\Delta = 1$ coordinates is modified. As can be seen from the pseudo-codes, the loop in line (4) iteratively scans the dimensions, whereas the loop in the line (5) modifies the values within the given dimension and then compares and retrieves the cases. Intuitively, limiting the retrieval to cases having up to $\Delta = 1$ modified coordinates is motivated by the observation that a higher number of modified coordinates would typically mean a lower similarity of cases. However, in certain conditions, e.g., retrieving a large number of cases with a low threshold, the retrieval with up

[5] The number of different values may be smaller due to the hashing collisions.

```
Retrieve (TARGET-CASE, K)
(1)  map TARGET-CASE to the hypercube of dimension $n$
(2)  assume the location of TARGET-CASE is $(h_1,…,h_n)$
(3)  let RETRIEVED-CASES be a set of size $K$, initially empty
(4)  for each dimension $i$ from $1$ to $n$
(5)    for each $x$ in the range of values for coordinate $i$
(6)      let CURRENT be the set of cases stored in
         the location $(h_1,…,h_{i-1},x,h_{i+1},…,h_n)$
(7)      for each TEST-CASE $\in$ CURRENT
(8)        compute similarity(TARGET-CASE, TEST-CASE)
(9)          insert TEST-CASE into RETRIEVED-CASES s.t.
             RETRIEVED-CASES is sorted according to the
             similarity values of the cases
(10) return RETRIEVED-CASES
```

Fig. 6 Algorithm for retrieving K most similar cases

to $\Delta = 1$ modified coordinates may not find a sufficient number of cases. In these conditions there is a need for a deeper retrieval, where the values of $\Delta > 1$ coordinates are modified. To adapt the proposed algorithm, the loop in the line (4) of the pseudo-code is replaced by a nested loop, which allows the values of multiple coordinates to be modified and retrieves cases with a higher number of modified coordinates.

The number of comparisons in the exhaustive retrieval is $O(N)$, where N is the total number of cases in the case-base. The complexity of UNSO-based retrieval is considerably lower, as the target case is compared only with the nearby cases in the MLH. The complexity of the UNSO-based retrieval for $\Delta = 1$ is $O(nk)$, where n is the number of dimensions in the MLH and k is the number of different coordinate values within these dimensions (assuming that it is constant).[6] In the general case of n features in the target case, Δ modified coordinates, and k possible values for each feature, the number of comparisons will be:

$$O(k^{\Delta} C^{n}_{\Delta}) = O\left(k^{\Delta} \frac{n!}{\Delta!(n-\Delta)!}\right)$$

Anyway, the number of comparisons in UNSO-based retrieval is bounded by the number of comparisons in the exhaustive retrieval, and this occurs when Δ is equal to the dimension n.

In summary, UNSO maintains the case-base as a distributed MLH graph. Grouping of similar cases facilitates retrieval of similar cases by an iteratively expanding retrieval. The proposed algorithm allows efficient and accurate retrieval of similar cases and distributes the required computational effort between the MLH nodes.

8 Empirical demonstration

To demonstrate validity of the proposed retrieval algorithm we conducted a study with 5 corpora of real-life E-Commerce ads from the following domains: *refrigerators*, *cameras*, *televisions*, *printers* and *mobile phones* (in short *mobiles*). The ads were downloaded from http://www.recycler.com and manually converted to unspecified vectors. For example, ad “Philips 50FD995 50” plasma TV, brand new, “$4000” was converted to <*price:4000, manufacturer:Philips, model:50FD995, size:50, screen:plasma, condition:new, package:true*>. The conversions were performed as closely as possible to the original content of the ads.

6 Practically, a smaller number of comparisons will be performed, as the MLH is sparse.

Table 2 Distribution of features in the corpora

Corpus	Cases	Features				Pairs				Pairs (%)		
		Bool.	Num.	FT	Total	Bool.	Num.	FT	Total	Bool.	Num.	FT
Refrigerators	61	2	4	4	10	22	117	115	254	8.6	46.1	45.3
Cameras	65	2	5	6	13	5	123	132	260	1.9	47.3	50.8
Televisions	76	1	3	7	11	3	148	135	286	1.0	51.8	47.2
Printers	94	5	2	4	11	48	100	293	441	10.9	22.7	66.4
Mobiles	130	3	1	5	9	70	130	383	583	12.0	22.3	65.7

Table 2 presents the following statistical properties of the corpora: number of cases in the corpus (‘cases’ column), number of different features for every type (‘features’ column and sub-columns ‘Boolean’, ‘Numeric’, ‘free-text’, and ‘total’), number of pairs, i.e., the frequency of the $feature_i$:$value_i$ pairs for every type (‘Pairs’ column, and sub-columns ‘Boolean’, ‘Numeric’, ‘free-text’, and ‘total’), and the percentage of the $feature_i$:$value_i$ pairs for every type [‘Pairs (%)’ column, and sub-columns ‘Boolean’, ‘Numeric,’ ‘free-text’, and ‘total’]. Note that the number of pairs indicates the degree of incompleteness of cases in the corpus. For example, consider the corpus of refrigerators. If every case was described by all the features, the corpus should have in total $61^*(2 + 4 + 4) = 610$ pairs. However, it has only 254 pairs.

We also analyzed statistical properties of individual features in the corpora. Table 3 presents the following statistical properties of the corpora: type and the name of the feature, number of occurrences in the corpus (denoted by o_f), number of different values (denoted by v_f), and the total number of occurrences and values for the features from every type (the last row *total*).

An UNSO-based model for storage and retrieval of cases was implemented in Java programming language. Every corpus of ads was assigned a separate MLH. Version 2.0 of WordNet was used to standardize the ads. The number of dimensions in the MLH was not limited, i.e., it was equal to the number of different features that appeared in their ads (shown in Table 2). The number of coordinate values that we used for each dimension, further referred to as cardinality (in short *card*), was 7.

On the one hand, low values of *card* may increase the probability of hashing collisions and hamper retrieval capabilities of the system, as it may retrieve irrelevant cases located in nearby nodes due to hashing collisions rather due to their similarity with the target case. Also, low values of *card* may skew the distribution of cases over the nodes and lead to unbalanced computational overheads. On the other hand, high values of *card* may generate large and sparse MLHs, which would be hard to maintain in a pure decentralized P2P environment. The $card = 7$ was selected to keep the MLH small and to allows reasonably good retrieval capabilities (Ben-Asher and Berkovsky 2006).

8.1 Grouping of similar cases

This experiment was aimed at validating the grouping property of UNSO-based MLH, i.e., mapping of similar cases to nearby MLH locations. Since no distance metric for case locations was defined for the MLH, we quantify the distances between two cases by the number of modified coordinates in their locations.

Table 3 Distribution of features and values from different types in the corpora

Corpus	Refrigerators			Cameras			Televisions			Printers			Mobiles		
	name $name_f$	o_f	v_f	$name_f$	o_f	v_f	$name_f$	o_f	v_f	$name_f$	o_f	v_f	$name_f$	o_f	v_f
Boolean	Delivery	6	2	Package	3	2	Package	3	2	Cable	5	2	Charger	51	2
	Ice-maker	16	2	Video	2	2				Ink	19	2	Manual	14	2
										Manual	6	2	SIM	5	2
										Software	7	2			
										Toner	11	2			
	Total	22	2	Total	5	2	Total	3	2	Total	48	2	Total	70	2
Numeric	Age	11	6	Memory	8	4	Price	76	51	Age	6	2	Price	130	45
	Price	61	28	Price	65	37	Size	70	20	Price	94	38			
	Size	38	25	Resolution	12	7	Produced	2	2						
	Warranty	7	2	Warranty	2	2									
				Zoom	36	12									
	Total	117	61	Total	123	62	Total	148	73	Total	100	40	Total	130	45
Free-text	Color	23	8	Body	4	4	Color	5	4	Condition	62	14	Case	17	4
	Condition	39	9	Condition	28	10	Condition	42	11	Manufact.	91	13	condition	75	13
	Manufact.	50	15	Flash	6	4	Manufact.	68	21	Model	85	79	Manufact.	129	16
	Model	3	3	Focus	6	2	Material	11	1	Type	55	4	Model	121	63
				Manufact.	63	22	Model	5	5				Network	41	3
				Type	25	5	Ratio	3	2						
							Type	1	1						
	Total	115	35	Total	132	47	Total	135	45	Total	293	110	Total	383	99

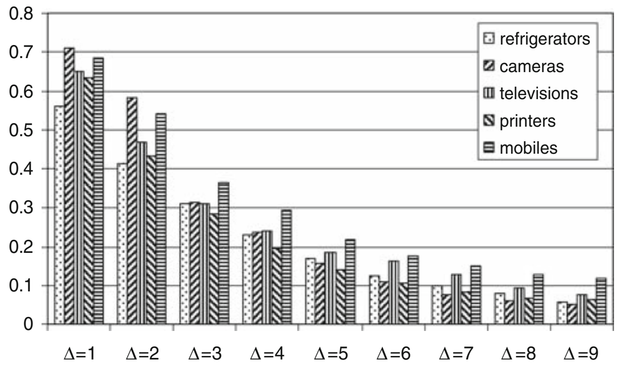


Fig. 7 Average similarity of cases versus the values of Δ

In every experiment, one case was selected to be the target case. For each of the other cases in the case-base, the number of modified coordinates Δ and the similarity of the cases were computed. The experiment was repeated for each case in the corpora being the target case. The similarity of cases as a function of Δ was computed by averaging the similarity of all the cases having Δ modified coordinates. Figure 7 illustrates the similarity of cases as a function of Δ in the above five corpora and Fig. 8 the distribution of cases over the values of Δ. In both figures Δ ranges from $\Delta = 1$ to $\Delta = 9$.

Figure 7 shows that the average similarity of cases monotonically decreases with the number of modified coordinates Δ. This validates our assumption regarding the grouping of similar cases in UNSO-based MLH, as the cases with a small number of modified coordinates are more similar to the target case than the cases with a large number of modified coordinates. Although similarity values differ across the corpora (the reasons will be discussed later in this chapter), the average similarity of cases monotonically decreases with Δ for every corpus. However, we should note that this observation is indicative only, as the similarity of cases depends not only on Δ, but also on the features and values that appear in the cases. For example, consider the cases: $c_1 = \ < manufacturer : BMW,\ doors : 3 >$, $c_2 = \ < manufacturer : Ferrari,\ doors : 3 >$, $c_3 = \ < produced : 2000,\ doors : 3 >$, and $c_4 = \ < produced : 2001,\ doors : 4 >$. Assuming that there are no hashing collisions, the number of modified coordinates between c_1 and c_2 is $\Delta = 1$, while between c_3 and c_4 it is $\Delta = 2$. However, the similarity of c_1 and c_2 is lower than the similarity of c_3 and c_4 due to the different types of the features.

Figure 8 shows that the distribution of cases over the values of Δ demonstrates a bell-curve distribution for all the corpora. There is a certain difference between the corpora, as the distribution of the televisions corpus is shifted towards the low values of Δ. This means that the percentage of cases with a lower number of modified coordinates in this corpus is higher that in the other corpora, i.e., television cases are grouped better than the cases in other corpora. The reasons for this behavior will be explained by analyzing statistical properties of the corpora. This characteristic of the televisions corpus will affect the performance of the proposed retrieval algorithm, as will be discussed later in this chapter.

8.2 Retrieval capabilities

This experiment was aimed at evaluating the accuracy of the proposed UNSO-based retrieval. For this, we retrieved the set of cases using the threshold-based retrieval (recall experiments,

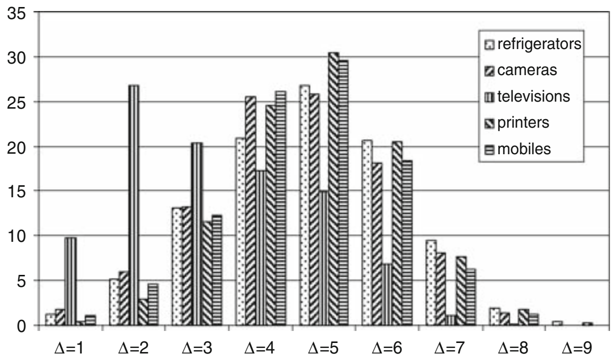


Fig. 8 Percentage of cases versus the values of Δ

Sect. 8.2.1) and the K most similar cases retrieval (precision experiments, Sect. 8.2.2), and then compared the retrieved set with the true set of the most similar cases retrieved by the exhaustive search. Hence, in every experiment one case was considered as the target case and two sets were retrieved: (1) the true set of the most similar cases R_e retrieved using the exhaustive search, and (2) the approximated set of the most similar cases R_u retrieved using the proposed UNSO-based search. Then, the retrieved sets R_e and R_u were compared. This process was repeated for each available case considered as the target case, i.e., number of times equal to the number of cases in the corpus. The overall value of recall/precision was computed by averaging the values obtained for every target case.

8.2.1 Recall

The accuracy of the threshold-based retrieval was evaluated using the Information Retrieval metric of recall (Salton and McGill 1983). Assuming that the set R_e contains all the relevant cases and the set R_u contains all the retrieved cases, the recall was computed as the ratio between the cardinality of the latter and the cardinality of the former:

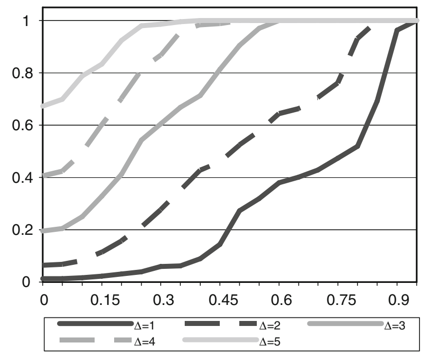
$$recall = \frac{|R_u|}{|R_e|}$$

Since R_u may only be a subset of R_e, the recall will always be lower or equal 1. There was no need to check the real relevance of cases, as in every experiment all the cases belong to one application domain.[7]

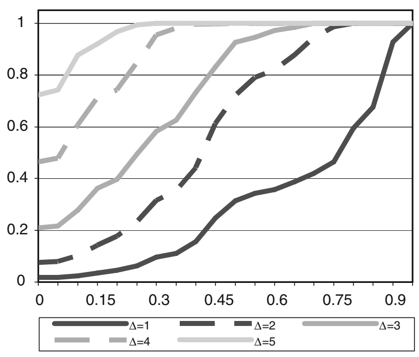
Figure 9 illustrates the values of the recall as a function of the similarity threshold β and the maximal number of modified coordinates Δ for all the five corpora. The horizontal axis represents the values of β and the vertical represents the recall. For each corpus we plot graphs for $\Delta = 1, 2, 3, 4$, and 5. The results show that the recall is correlated with the number of modified coordinates Δ. For low Δ, the number of cases found by UNSO-based retrieval is low and (since R_e is unchanged) the recall is low. Increasing Δ expands the search space, the retrieved set R_u gets closer to the real set R_e, the recall increases, and converges faster to the maximal value of 1. This observation is true for all the corpora.

For a low threshold β, i.e., retrieved cases are not necessarily highly similar to the target, and for low Δ the recall in most corpora is low. This is explained by the observation that cases having low similarity with the target case may be located not only in nearby, but also in further locations, and may have many different feature values. As such, the proposed

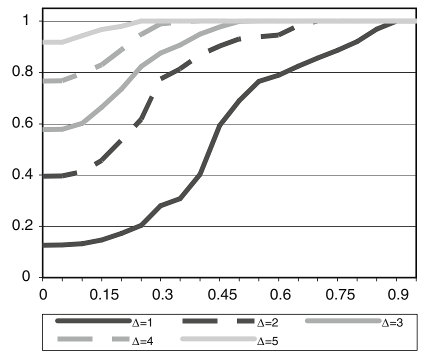
[7] In this setting $precision = 1$ as all the retrieved cases have the required similarity degree.



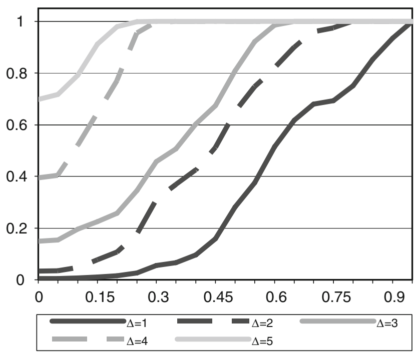
(a)



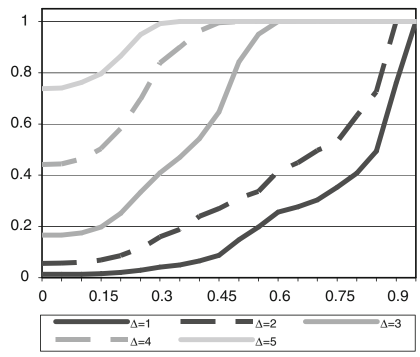
(b)



(c)



(d)



(e)

Fig. 9 Recall of the retrieval versus the similarity threshold β for different values of Δ. **a** Refrigerators. **b** Cameras. **c** Televisions. **d** Printers. **e** Mobile phones

retrieval may not find these cases and the recall is low.[8] When β increases, the cases with a low similarity are excluded both in R_u and in R_e and the recall increases. Hence, for high β, the two sets are very close and the recall converges to 1. This means that for high values of

[8] This issue is of a minor importance in real-life CBR systems, in which the retrieval is aimed at finding cases having a high similarity with the target case.

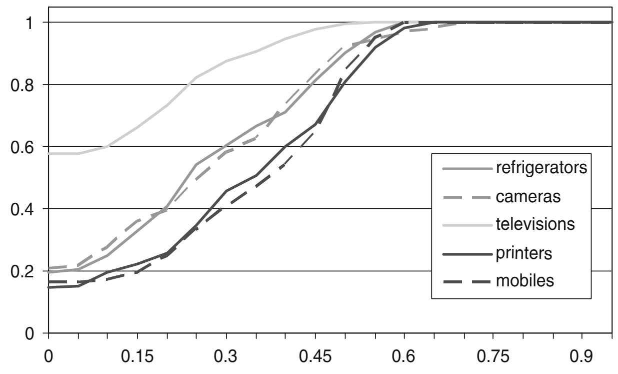


Fig. 10 Recall in different corpora versus similarity threshold β for $\Delta = 3$

β the set of cases retrieved by the UNSO-based approach is very similar to the set of cases retrieved by the traditional exhaustive approach. Note that measuring the precision in this setting is meaningless (Salton and McGill 1983), as all the cases in R_u have similarity higher than the similarity threshold β and the precision is 1.

Although the recall increases with β and converges to 1 faster with the increase of Δ, the behavior of recall differs across the five corpora. The most noticeable difference is observed for low β, where the proposed retrieval algorithm considers only a small portion of potentially similar cases. Hence, many cases are missed and the recall is low. To compare between the five corpora we plot in Fig. 10 the recall curves for different corpora as a function of β for $\Delta = 3$.

The figure shows that although the behavior of recall in all the corpora is similar, the recall curves of the five corpora can be partitioned into three groups. The first group includes the corpus of refrigerators, which demonstrates the highest recall. The second group includes the corpora of cameras and televisions, which demonstrate moderate recall (the recall values are quite similar). The third group includes the corpora of printers and mobile phones, which demonstrate the lowest recall (also very similar). To explain these differences, we will present later in this section an elaborate analysis of the statistics of features and values in the corpora.

8.2.2 Precision

Also, we conducted experiments evaluating the precision the K most similar cases retrieval. Assuming that the set R_e contains all the relevant cases and the set R_u contains all the retrieved cases, the precision is computed as the proportion of the K real cases retrieved by the approximated UNSO-based retrieval:[9]

$$precision = \frac{|R_u \cap R_e|}{|R_u|} = \frac{|R_u \cap R_e|}{K}$$

There was no need to check the relevance of cases, as in every experiment all the cases belong to one application domain.

Figure 11 illustrates the values of the precision as a function of K and the maximal number of modified coordinates Δ for all the five corpora. The horizontal axis represents K and the

[9] Since the size of the retrieved set is limited to K, the computed metric is Precision@K rather than the standard precision (Salton and McGill 1983). Due to this we eschew the computation of recall.

vertical—the precision. For each corpus we plot graphs for $\Delta = 1, 2, 3, 4$, and 5. Similarly to the recall in the threshold-based retrieval, the precision of the K most similar cases retrieval is correlated with Δ. For low Δ, the number of cases found by UNSO-based retrieval is low (see low values of Δ in Fig. 8). Moreover, in many cases it is lower than the number of cases to be retrieved K. As such, high precision cannot be achieved regardless of the similarity of cases that are found. Increasing Δ expands the search space, the precision increases and gets closer to the maximal value of 1, especially for low K. This observation is true for all the corpora.

For low K (retrieval of highly similar cases), the precision is relatively high and the sets R_e and R_u are similar. For higher K the precision decreases. This is explained by the observation that for a certain Δ the set R_u does not change (always the cases having up to Δ modified coordinates) and the cardinality of these set may not be sufficient for a large K. It should be noted that practical CBR systems typically retrieve a small number of highly similar cases. Hence, the accuracy of the proposed retrieval should be evaluated at low K, where the precision is relatively high.

To compare the precision across the different corpora, we plot in Fig. 12 the curves of the precision as a function of K for $\Delta = 3$. Their behavior is similar and the precision decreases with K. Similarly to the recall curves, also the precision curves can be partitioned into 3 groups. The first group includes the corpus of refrigerators, which demonstrates the highest precision. The second group includes the corpora of cameras and televisions, which demonstrate moderate precision (the precision values are quite similar). The third group includes the corpora of printers and mobile phones, which demonstrate the lowest precision (also very similar). In the following subsection we will analyze statistical properties of the cases and the corpora to understand the differences in the retrieval capabilities (both the precision and the recall).

8.2.3 Analysis

Given a target case, the exhaustive retrieval compares it with all N cases stored in the case-base. Alternatively, the proposed retrieval algorithm considers only $M \leq N$ cases having up to Δ modified coordinates. The lower is the ratio M/N, the higher is the probability that recall and precision will not reach their optimal value 1. Practically, this may happen due to two reasons: (1) not enough cases considered, i.e., M is smaller than the number of cases having similarity above the required threshold or than the number of cases to be retrieved, or (2) the considered cases are not sufficiently similar, i.e., among the considered cases there are not enough cases with the required similarity or these cases are not similar to the extent that will allow them to be retrieved. Thus, low accuracy is expected when the number of cases to be retrieved is high: either K is high value or β is low. On the contrary, high accuracy is expected when retrieving a small number of similar cases, as those will typically have a small number of modified coordinates and will be found by the algorithm.

Although both the recall and precision are high for retrieving a small set of similar cases, certain differences between the corpora (and between the types of ads) do exist. To understand it, we analyzed the correlation between various statistical properties of the corpora and the performance of the algorithm. We identified two factors that may affect the retrieval capabilities:

- Average number of the pairs in a case, further denoted by $pairs_{case}$. It is computed by dividing the total number of pairs that appear in the cases of a given corpus by the number of cases in the corpus. Since the proposed algorithm iteratively modifies the values of up

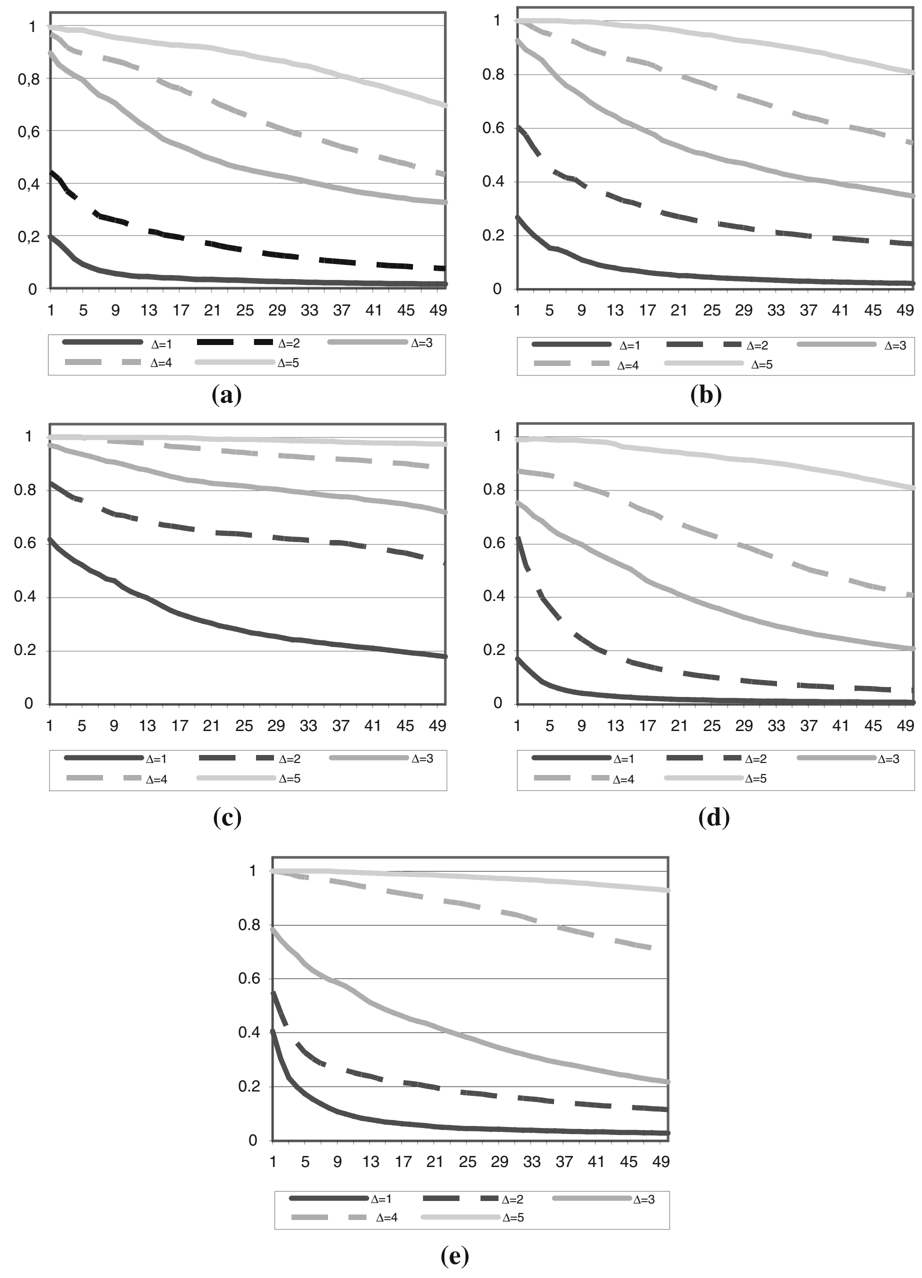


Fig. 11 Precision of the retrieval versus the number of retrieved cases K for different Δ. **a** Refrigerators. **b** Cameras. **c** Televisions. **d** Printers. **e** Mobile phones

to Δ coordinates, a low value of $pairs_{case}$ in a corpus allows the search to find more cases. In fact, if $pairs_{case}$ is low, then two arbitrary cases are described, on average, by a small number of different pairs. Hence, modifying a small number of coordinates in one case is likely to find the other case. As more cases are found by the algorithm, more candidate

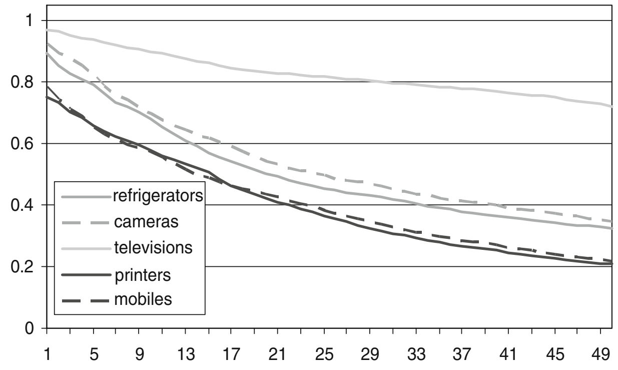


Fig. 12 Precision in different corpora versus number of cases to retrieve K for $\Delta = 3$

Table 4 Statistical properties of data in the corpora

Corpus	Refrigerators	Cameras	Televisions	Printers	Mobiles
$pairs_{case}$	4.164	4.000	3.763	4.692	4.485
var_{FT}	8.750	8.250	6.429	27.500	24.750
var_{Num}	0.262	0.262	0.235	0.292	0.308

cases are considered and compared, and precision and recall of the retrieval increase (as the true set of most similar cases is unchanged). Alternatively, if $pairs_{case}$ is high, a smaller number of cases having Δ modified coordinates will be found, and the retrieval capabilities will be hampered. As such, retrieval capabilities of the propose retrieval improve with the decrease of $pairs_{case}$.

- Variability of values of the features in a corpus, which indicates how different two values are expected to be. The variability is computed separately for different types of features, as it depends on the similarity metric being exploited. For example, consider the free-text and Boolean features, for which the similarity is computed by exact matching, the variability var_{FT} is correlated with the number of possible values of a feature. Conversely, for the Numeric features, for which the similarity depends on the difference between the values and the maximal range of values, the variability var_{Num} is correlated with the variance of values of a feature. When the variability is low, the retrieval of cases having Δ modified coordinates will find a larger number of cases. Since the set of cases retrieved by the exhaustive search is unchanged, both the precision and the recall will increase. Alternatively, when the variability is high, the organization of cases is sparse and the retrieval capabilities are hampered.

Table 4 shows the values of $pairs_{case}$, var_{FT} and var_{Num} in all the corpora.[10] var_{FT} of a corpus was computed as a weighted according to o_f average of the v_f for all the features (see Table 3). var_{Num} was computed as a weighted average of the numeric variability, i.e., standard deviation of features divided by their maximal difference.

We computed the correlation between these factors and the averaged for all the available values of β recall of the proposed algorithm. The recall is negatively correlated with all the factors. For example, for $\Delta = 3$ the correlation with $pairs_{case}$ is $-0.882 (p = 0.024)$, with var_{FT} it is $-0.765 (p = 0.066)$, and with var_{Num} it is $-0.944 (p = 0.008)$. Note that 2 out

[10] The variability of Boolean features is not computed as they have two values: *true* or *false*.

of 3 correlations are statistically significant ($p < 0.05$). Similar correlations were observed also for other values of Δ Even qualitatively, $pairs_{case}$, var_{FT}, and var_{Num} of the televisions corpus are lowest, whereas the precision and the recall in this corpus are highest. For the refrigerators and cameras corpora, $pairs_{case}$, var_{FT}, and var_{Num} are slightly higher and the retrieval capabilities are worse. Finally, for the printers and mobile phones corpora, $pairs_{case}$, var_{FT}, and var_{Num} are considerably higher, whereas their retrieval capabilities are worst. These correlations between the statistical characteristics of the corpora and the values of the recall and precision explain the differences in the retrieval capabilities of the proposed retrieval algorithm.

It worth noting that when applying the proposed retrieval algorithm, the maximal number of modified coordinates Δ should be adapted to the value of $pairs_{case}$. For example, consider the precision of the K most similar cases retrieval and assume that we are interested in a precision greater than 0.8 for retrieving $K = 5$ most similar cases. Figure 11 shows that for the televisions corpus with a low $pairs_{case}$, a search with $\Delta = 2$ modified coordinates will suffice. On the other hand, for refrigerators and cameras $pairs_{case}$ is higher and a search with $\Delta = 3$ modified coordinates has to be conducted. Finally, for the printers and mobile phones with even higher $pairs_{case}$, a deeper search with $\Delta = 4$ modified coordinates is required to obtain the same accuracy of the results.

8.3 Computational optimization

The main goal of the proposed algorithm is to decrease the number of case comparisons, while maintaining the quality of the results. This reduces the required computational effort due to the fact that the target case is compared only with cases having up to Δ modified coordinates, rather than with all the cases stored in the case-base. Since the cases are stored distributively, the comparisons are performed at the user-side and do not involve any central processing. This resolves the computational bottleneck of central processing and allows additional spreading of the computational effort.

This experiment was aimed at evaluating the number of case comparisons in the proposed algorithm. In each execution of the experiment, a single target case was considered, the sets of the most similar cases were retrieved using the threshold-based retrieval, and the number of comparisons performed was computed. The experiment was repeated for all the cases in the case-base. The number of comparisons was computed by averaging the numbers of comparisons for each chosen target case. Table 5 shows the average number of comparisons (denoted by *comp*) in different corpora for the maximal number of modified coordinates ranging from $\Delta = 1$ to $\Delta = 5$. Note that the number of comparisons does not depend on β or K, but only on Δ. This number should be compared to with the number of comparisons in the exhaustive retrieval (denoted by *exh*), which is equal to the number of cases in the corpus minus 1. To analyze the results, we computed the relative number (percentage) of comparisons that were performed (denoted by %) by dividing *comp* by *exh*.

The results for all the corpora show that although the number of case comparisons in the proposed retrieval steadily increases Δ, it remains lower than the number of comparisons in the exhaustive retrieval. This is explained by the fact that for higher Δ the retrieval is expanded and more cases are compared. However, even for $\Delta = 5$ the number of comparisons is still lower than in the exhaustive retrieval. The results also show that for any Δ, the relative number of comparisons in the refrigerators, cameras, printers and mobile phones corpora is lower than in the televisions corpus. This is explained by the smaller $pairs_{case}$ of the television cases leading to their better grouping in the MLH. This is also supported

Table 5 Number of comparisons in the approximated retrieval

Corpus	*exh*	$\Delta = 1$		$\Delta = 2$		$\Delta = 3$		$\Delta = 4$		$\Delta = 5$	
		Comp	%	Comp	%	Comp	%	Comp	%	Comp	%
Refrigerators	60	0.92	1.53	4	6.67	11.84	19.73	24.39	40.66	40.46	67.43
Cameras	64	1.16	1.83	4.98	7.79	13.45	21.01	29.78	46.54	46.31	72.36
Televisions	75	10.13	13.51	30.24	40.52	43.79	58.39	57.76	77.02	68.92	91.89
Printers	93	0.45	0.48	3.15	3.39	13.85	14.89	36.66	39.42	64.87	69.76
Mobiles	129	1.74	1.35	7.71	5.98	21.57	16.72	57.29	44.41	95.45	73.99

by Fig. 8, showing that television cases generate a denser structure than the cases in other corpora for a low Δ.

Revisiting the above example retrieval of $K = 5$ most similar cases with precision greater than 0.8, we note the following computational gain. Search with $\Delta = 2$ modified coordinates in the dense televisions corpus will compare the target case with approximately 40% of the cases in the case-base, search with $\Delta = 3$ in the refrigerators and cameras corpora will compare approximately 20% of the cases, while the most expanded search with $\Delta = 4$ in the printers and mobile phones corpora will compare approximately 40–45% of cases. Hence, even for these quite restricting conditions the proposed retrieval algorithm considerably decreases the computational overhead of the exhaustive retrieval.

This experiment shows the trade-off between the retrieval capabilities of the proposed retrieval and the computational optimization achieved. The highest recall and precision of the televisions corpus are obtained at the expense of a larger number of case comparisons performed in the retrieval. For the other corpora, the recall and precision are lower; however, also the number of case comparisons is lower. In summary, combining the outcomes of the experiments presented in Sects. 8.2 and 8.3 allows us to conclude that the proposed retrieval algorithm (1) decreases the number of comparisons performed when retrieving the most similar cases, while (2) keeps reasonably good recall and precision with respect to the set of cases retrieved using the exhaustive retrieval.

9 Conclusions and future research

This work reviewed the current state of the art in P2P and distributed CBR and presented a novel approach to pure decentralized P2P storage of CBR cases using a multi-layered hypercube (MLH) graph built using the UNSpecified Ontology (UNSO). UNSO facilitates considerably free descriptions of the cases, as the cases are described using a list of $feature_i$:$value_i$ pairs, where neither the features nor the respective values are restricted by any predefined ontology. The observation that the MLH inherently groups similar cases facilitated the development of an approximated algorithm for retrieval of most similar cases. This algorithm can be described as a localized search among cases located nearby the target case, as these cases are supposed to be similar to the target case.

Retrieval experiments, conducted over five corpora of real-life E-Commerce advertisements exemplified that the proposed retrieval algorithm accurately retrieves the most similar cases. Retrieval capabilities of the proposed approach are good for both the K most similar and threshold-based retrieval policies. The sets of cases retrieved by the exhaustive and the proposed retrievals are very similar for the retrieval of highly similar cases. In addition, the required computational effort, i.e., the number of comparisons, decreased considerably. Practically, most real-life CBR applications aim at retrieving a low number of very similar cases. Hence, the good performance demonstrated by the proposed approach in these settings is especially important. In this work we also conducted an elaborate analysis of the statistical properties of the data that affect the performance of the proposed approach. This analysis allowed us to draw conclusions regarding the specific conditions, in which the proposed retrieval algorithm is highly beneficial, as it provides both accurate (in terms of the precision and recall) and efficient (in terms of the number of comparisons) alternative to the exhaustive retrieval. To further improve the retrieval, its parameters should be adapted to the statistical properties of the data in the case-bases.

Although we assumed that the MLH is constructed in a pure distributed P2P environment, in this work we implemented its centralized prototype. Hence, we were not able to measure the improvement achieved due to the distribution of the case-base. In the future, we plan to implement a decentralized version of the MLH and to conduct another evaluation. Also, the experiments were conducted on relatively small corpora having at most 130 cases. This may raise questions regarding the statistical validity of the results. To provide solid experimental evidence, we plan to build larger case-bases and to conduct an elaborate experimental evaluation.

References

Aamodt A, Plaza E (1994) Case-based reasoning: foundational issues, methodological variations, and system approaches. AI Commun 7(1):39–59

Adar E, Huberman B (2000) Free riding on Gnutella. Technical report, Xerox PARC

Androutsellis-Theotokis S, Spinellis D (2004) A survey of peer-to-peer content distribution technologies. ACM Comput Surv 36(4):335–371

Bawa M, Condie T, Ganesan P (2005) LSH forest: self-tuning indexes for similarity search. In: Proceedings of the international conference on World Wide Web, Japan

Ben-Asher Y, Berkovsky S (2006) UNSO: unspecified ontologies for peer-to-peer E-commerce applications. J Data Semant 6:115–142

Bernstein A, Kaufmann E, Buerki C, Klein M (2005) How similar is it? Towards personalized similarity measures in ontologies. In: Proceedings of the internationale Tagung Wirtschaftsinformatik, Germany

Bogaerts S, Leake D (2004) Facilitating CBR for incompletely-described cases: distance metrics for partial problem descriptions. In: Proceedings of the European conference on case-based reasoning, Spain

Bonifacio M, Bouquet P, Mameli G, Nori M (2003) Peer-mediated distributed knowledge management. Agent-Mediated Knowledge Management, Springer

Branting K, Aha DW (1995) Stratified case-based reasoning: reusing hierarchical problem solving episodes. In: Proceedings of the international joint conference on artificial intelligence, CA

Clarke I, Sandberg O, Wiley B, Hong T (2000) Freenet: a distributed anonymous information storage and retrieval system. In: Proceedings of the workshop on design issues in anonymity and unobservability, CA

Coyle L, Doyle D, Cunningham P (2004) Representing similarity for CBR in XML. In: Proceedings of the European conference on advances in case-based reasoning, Spain

de Mantaras RL, McSherry D, Bridge D, Leake D, Smyth B, Craw S, Faltings B, Maher ML, Cox MT, Forbus K, Keane M, Aamodt A, Watson I (2005) Retrieval, reuse, revision and retention in case-based reasoning. Knowl Eng Rev 20(3):215–240

Fellbaum C (1998) WordNet–an electronic lexical database. MIT Press Publishers

Friedman JH, Bentley JH, Finkel RA (1977) An algorithm for finding best matches in logarithmic expected time. ACM Trans Math Softw 3(3):209–226

Gruber TR (1993) A translation approach to portable ontology specifications. Knowl Acquis J 6(2):199–220

Harren M, Hellerstein JM, Huebsch R, Loo BT, Shenker S, Stoica I (2002) Complex queries in DHT-based peer-to-peer networks. In: Proceedings of the international workshop on peer-to-peer systems, MA

Herlocker JL, Konstan JA, Borchers A, Riedl J (1999) An algorithmic framework for performing collaborative filtering. In: Proceedings of the international SIGIR conference on research and development in information retrieval, CA

Kalnis P, Ng WS, Ooi BC, Tan KL (2006) Answering similarity queries in peer-to-peer networks. Inf Syst J 31(1):57–72

Leake DB, Sooriamurthi R (2001) When two case bases are better than one: exploiting multiple case bases. In: Proceedings of the international conference on case-based reasoning, Canada

Leake DB, Sooriamurthi R (2003) Dispatching cases versus merging case-bases: when MCBR matters. In: Proceedings of the international Florida artificial intelligence research society conference, FL

McGinty L, Smyth B (2001) Collaborative case-based reasoning: applications in personalised route planning. In: Proceedings of the international conference on case-based reasoning, Canada

Milojicic DS, Kalogeraki V, Lukose R, Nagaraja K, Pruyne J, Richard B, Rollins S, Xu Z (2002) Peer-to-peer computing. Technical report HPL-2002-57, HP Labs

Nagendra Prasad MV, Lesser V, Lander S (1996) Retrieval and reasoning in distributed case bases. J Vis Commun Image Represent, Special Issue on Digital Libraries 7(1):74–87

Napster Inc, The Napster homepage. http://www.napster.com

Plaxton C, Rajaraman R, Richa A (1997) Accessing nearby copies of replicated objects in a distributed environment. In: Proceedings of the symposium on parallel algorithms and architectures, RI

Plaza E, McGinty L (2005) Distributed case-based reasoning. Knowl Eng Rev 20(3):261–265

Plaza E, Arcos JL, Martin F (1996) Cooperative case-based reasoning. In: Proceedings of the workshop distributed artificial intelligence meets machine learning, Hungary

Pouwelse J, van Slobbe M, Wang J, Reinders MJT, Sips H (2005) P2P-based PVR recommendation using friends, taste buddies and superpeers. In: Proceedings of the beyond personalization workshop, CA

Ratnasamy S, Francis P, Handley M, Karp R, Shenker S (2001) A scalable content-addressable network. In: Proceedings of the conference of the special interest group on data communication, CA

Richter MM (1992) Classification and learning of similarity measure. In: Proceedings of the annual conference of the German society for classification, Germany

Rowstron A, Druschel P (2001) Pastry: scalable, distributed object location and routing for large-scale peer-to-peer systems. In: Proceedings of the international conference on distributed systems platforms, Germany

Salton G, McGill M (1983) Introduction to modern information retrieval. McGraw-Hill Publishing

Schlosser M, Sintek M, Decker S, Nejdl W (2002) A scalable and ontology-based P2P infrastructure for semantic web services. In: Proceedings of the international conference on peer-to-peer computing, Sweden

Smyth B, Cunningham P (1996) The utility problem analysed: a case-based reasoning perspective. In: Proceedings of the European workshop on case-based reasoning, Switzerland

Tanin E, Nayar D, Samet H (2005) An efficient nearest neighbor algorithm for P2P settings. In: Proceedings of the national conference on digital government research, GA

Tran DA (2005) Hierarchical semantic overlay approach to P2P similarity search. In: Proceedings of the USENIX annual technical conference, CA

Watson I (1997) Applying case-based reasoning: techniques for enterprise systems. Morgan Kaufmann Publishers

Watson I, Gardingen D (1999) A distributed case-based reasoning application for engineering sales support. In: Proceedings of the international joint conference on artificial intelligence, CA

Wess S, Althoff KD, Derwand G (1993) Using K-d trees to improve the retrieval step in case-based reasoning. In: Proceedings of the European workshop on case-based reasoning, Germany

Wilson DR, Martinez TR (1997) Improved Heterogeneous distance functions. J Artif Intell Res 6:3–21

Wooldridge M (2002) An introduction to multi-agent systems. John Wiley Publishers