Semantic Peer, Here are the Neighbors You Want!*

Stefano Lodi, Wilma Penzo
DEIS
University of Bologna, Italy
{firstname.lastname}@unibo.it

Federica Mandreoli, Riccardo Martoglia, Simona Sassatelli
DII
University of Modena and Reggio Emilia, Italy
{firstname.lastname}@unimo.it

ABSTRACT

Peer Data Management Systems (PDMSs) have been introduced as a solution to the problem of large-scale sharing of semantically rich data. A PDMS consists of semantic peers connected through semantic mappings. Querying a PDMS may lead to very poor results, because of the semantic degradation due to the approximations given by the traversal of the semantic mappings, thus leading to the problem of how to boost a network of mappings in a PDMS.

In this paper we propose a strategy for the incremental maintenance of a flexible network organization that clusters together peers which are semantically related in Semantic Overlay Networks (SONs), while maintaining a high degree of node autonomy. Semantic features, a summarized representation of clusters, are stored in a "light" structure which effectively assists a newly entering peer when choosing its semantically closest overlay networks. Then, each peer is supported in the selection of its own neighbors within each overlay network according to two policies: Range-based selection and k-NN selection. For both policies, we introduce specific algorithms which exploit a distributed indexing mechanism for efficient network navigation. The proposed approach has been implemented in a prototype where its effectiveness and efficiency have been extensively tested.

1. INTRODUCTION

In recent years, Peer-to-Peer (P2P) systems have known an enormous success among Internet users. In these systems, users, called peers, connect each other for data sharing purposes. Notable examples are Napster, Kazaa, and Gnutella, just to mention a few.

On the other hand, as envisioned by the Semantic Web [3], the need of complementing the Web with more semantics has spurred much efforts towards a rich representation of data, giving rise to the widespread use of ontologies, XML schemas, RDF schemas, aso.

*This work is partially supported by the Italian Council cofunded project NeP4B

EDBT'08, March 25–30, 2008, Nantes, France.

In this view, Peer Data Management Systems (PDMSs) have been introduced as a solution to the problem of large-scale sharing of semantically rich data [10]. In a PDMS, peers are autonomous and heterogeneous data sources, having their own content modeled upon schemas. Because of the absence of common understanding of the vocabulary used at each peer's schema, semantic relationships are established locally between peers, thus implementing a decentralized schema mediation.

Being peers given more semantics, new potentialities are available as to query formulation and, consequently, new challenges arise for query processing. In a PDMS, queries are answered globally on a network of semantically related peers: The key issue is that peer mappings are specified locally and answering a query posed at a peer may require piecing together multiple peer mappings to locate the relevant data. Query reformulation is thus a challenging task, since the ability to obtain relevant data from other nodes in a network depends on the existence of a semantic path from the queried peer to that node. The semantic path needs to relate the terms used in the query with the terms specified by the node providing the data. Hence it is likely that there will be information loss along long paths in the PDMS because of missing (or incomplete) mappings, leading to the problem of how to boost a network of mappings in a PDMS [10].

Thus, for a given peer, the linkage closeness to semantically similar peers is a crucial issue for query processing to be both efficient and effective. A similar requirement has already been evidenced by works on Semantic Overlay Networks (SONs)[6] for P2P systems, which propose to cluster together peers with semantically similar content to improve the efficiency of query answering. However, in the context of a PDMS, this need is even more pressing and challenging because of the heterogeneity of the vocabularies used at the peers' schemas, thus also arising effectiveness issues as to the relevance of results.

The work presented in this paper aims to support the creation and maintenance of a flexible network organization for PDMSs that clusters together heterogeneous peers which are semantically related. The approach we propose is

- scalable, in the way it gracefully copes with the changes occurring in the network;
- incremental and self-adaptive, in the creation and maintenance of SONs as semantic clusters of peers;
- effective and efficient, in the way it supports the interoperability of a set of fully autonomous peers interconnected through semantic mappings.

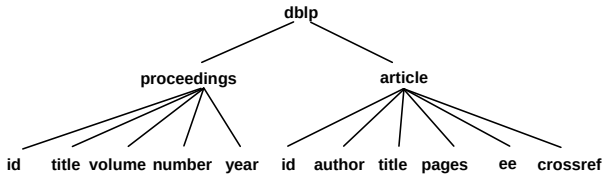


Figure 1: DBLP schema

More precisely, we present a mechanism of *semantic clustering* which assists each newly entering peer in the selection of its neighbors in a two-fold fashion: First, in a *coarse-grained* choice of the semantically closest overlay networks[1]; Then, within each overlay network, in a *fine-grained* selection of its own neighbors among its most semantically related peers.

The first step relies on a scalable structure which abstracts out the essential features about the SONs available in the network for an effective selection. This structure relies on principles similar to the ones presented in [9] for clustering large databasets and it incrementally evolves as new peers enter the system.

An accurate choice of the neighbors is then supported through the introduction of two kinds of selections, one based on a semantic similarity threshold (*range-based selection*) and the other one based on a maximum number of neighbors (*k-NN selection*). Instead of adopting a broadcast-based approach, we provide an efficient support to both solutions by introducing algorithms which prune out non-relevant peers and avoid useless computations. Our approach draws inspiration from the M-tree algorithms [4]. In this way, we drastically lighten the network load needed for an optimal insertion of new peers into each SON. This is an essential aspect, as most of the PDMS resources are usually taken up to solve the queries spanning simultaneously the network.

The organization of the paper is as follows: Section 2 provides an overview of the approach which is then deepened in Sections 3, 4, and 5. In Section 6 we briefly discuss about the impact of the network organization we propose on query processing. Section 7 discusses the experiments we conducted on a prototype implementing the approach. Finally, in Section 8 we discuss literature and conclude with future research directions to be explored.

2. OVERVIEW

The network we refer to is made up of a set of semantic peers, each represented by a set of concepts $\{c_1, ..., c_m\}$ describing its main topics of interest. The process leading to the representation of each peer is out of the main scope of this paper. In principle, they derive from the peer's local schema as it describes the semantic content of the underlying data. To this end, solutions like the one recently proposed in [26] could be adopted. For instance, the concepts extracted from the portion of the DBLP XML schema depicted in Fig. 1 could be $\{proceedings, article, author\}$.

The network is organized in a set of Semantic Overlay Networks (SONs) [6] in such a way to assist each newly entering peer in the selection of the semantically closest peers as its neighbors. A SON is a group of semantically related nodes locally connected through a link structure. A sample of network made up by two SONs supporting a web of research-related data is shown in Fig. 2. It includes various peers. Some of them, such as the EDBT Association (EDBT Ass.) and the University of Rome (URome) are "monothematic", i.e. they only deal with publications and university people, respectively. Other peers, instead, are concerned with both themes, e.g. Stanford.

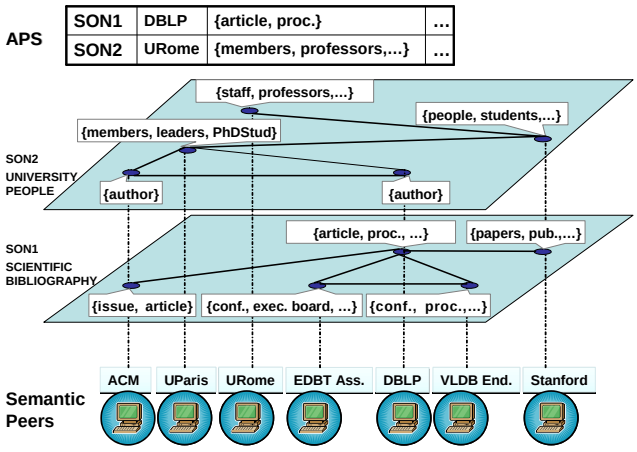


Figure 2: Sample of network organization

Peers are assigned to one or more SONs on the basis of their own concepts. In a PDMS, this operation is a really challenging one because of the lack of a common understanding among the peer's local dictionaries. This means that similar or even the same contents in different peers are not usually described by the same concepts. Our proposal is to solve such heterogeneity by clustering together in the same SON nodes with semantically similar concepts. Semantic similarity is also at the basis of the approach we propose to guide the selection of the neighbors within each SON.

The network evolves incrementally to assimilate entering peers. When a peer joins the system, it first performs a coarse-grained neighbor selection by accessing the *Access Point Structure (APS)*. This is a "light" structure which maintains cumulative information about the SONs available in the network. It is conceptually centralized but it can be stored in a distributed manner among different peers, for instance, by means of a Distributed Hash Table (DHT). The APS helps the newly entering peer to decide which SONs to join or whether to form new SONs by providing useful information such as SON's representative concepts (see the third column of the APS in Fig. 2) which the peer compares with its concepts.

After SON selection, the peer starts to navigate the link structure within each selected SON from the entry-point peer exemplified in the second column of the APS of Fig. 2. We assist the peer in the selection of the semantically closest peers by providing two policies: 1) Range-based, where the new peer connects to all the peers in a given semantic similarity range, and 2) kNN-based, where the k semantically closest peers are chosen. For both policies, we introduce efficient algorithms which exploit index structures distributed across the network to reduce the search space.

[1]This is very common in P2P communities, where peers share common interests, and a peer can belong to more than one SON, depending on its own interests.

3. ON DISTANCE DEFINITION

Similarities between pairs of concepts are captured by a distance function d. Several alternatives are possible for its implementation as we only require it is a metric, i.e. it satisfies the triangle inequality, the symmetry and the non-negativity properties. Although its definition is orthogonal to the proposed approach for SONs, the goodness of the obtained results are heavily dependent from its effectiveness. One of the best ways of exploiting the semantics of the involved concepts is to correlate them by adopting knowledge-based distances, which take advantage of linguistic information extracted from external knowledge sources. For this reason, in this paper we will consider two variants of two of the most effective distances in the Computational Linguistics field, exploiting the well-known and widely available WordNet (WN) thesaurus. Both measures quantify the distance between two given concepts c^i and c^j by comparing their WN hypernymy hierarchies. The first distance we consider, straightly derived from the one presented in [8], is obtained by computing the depths of the concepts in the WN hypernymy hierarchy and the length of the path connecting them as follows (GM distance in the following):

$$d(c^i, c^j) = 1 - \frac{2 * \text{depth of } lca(c^i, c^j)}{\text{depth of } c^i + \text{depth of } c^j},$$

where $lca(c^i, c^j)$ is the least common ancestor between c^i and c^j. The other option we consider for d is an adaptation of the Leacock-Chodorow [13] distance (LC distance in the following):

$$d(c^i, c^j) = \begin{cases} \frac{len(c^i, c^j)}{2 \cdot H} & \text{if } \exists\ lca(c^i, c^j) \\ 1 & \text{otherwise} \end{cases}$$

where $len(c^i, c^j)$ is the number of links connecting c^i and c^j in the hypernymy hierarchy, and H is the height of the hierarchy (16 in WordNet). Both distances take values between 0 (equal concepts or synonyms) and 1 (completely unrelated concepts) and satisfy all the required properties.

The distance function d is extended to sets of concepts. To this end, a function f should be defined which, given two sets of concepts C_i and C_j and the set of distances $d(c^i, c^j)$ between each concept $c^i \in C_i$ and $c^j \in C_j$, maps to a distance value $d(C_i, C_j)$ between the two sets. Such a function must preserve the metric properties. Moreover, we also require that, for a set of concepts $C = \{c_1, \ldots, c_n\}$ such that $d(c_i, C_j) < d(c_i, C_k)$ for each $i = 1, \ldots, n$, then $d(C, C_j) < d(C, C_k)$ (monotonicity). Among the possible alternatives, in the following we will adopt the function which defines $d(C_i, C_j)$ as the distance value $d(Cl_i, Cl_j)$ between the C_i's clustroid, Cl_i, and C_j's clustroid, Cl_j.

In order to define the notion of *clustroid* [9], the centrally located concept of a set of concepts, we first introduce the notion of distance factor.

DEFINITION 1 (DISTFACTOR). *Let $C = \{c_1, ..., c_n\}$ be a set of concepts. The $DistFactor(c)$ of a concept $c \in C$ is defined as $DistFactor(c_k) = \sum_{i=1}^{n} d^2(c, c_i)$.*

Then, the clustroid is defined as follows.

DEFINITION 2 (CLUSTROID). *Given a set of concepts C, the* clustroid *of C is defined as the concept $Cl \in C$ such that $\forall c \in C : DistFactor(Cl) \leq DistFactor(c)$.*

4. CHOOSING SONS

This section describes the initial actions performed by a newly entering peer.

4.1 APS: Access Point Structure

SONs are clusters of connected peers which share similar concepts and which evolve incrementally as new peers join the network. The APS ignores link structures and provides an abstraction of the SONs as *clusters of concepts* (e.g. $issue, article, conference$, etc. for SON1 in Fig. 2). In order for the APS to be a "light" structure which scales to the large, we do not keep all concepts at the APS level and follow an approach similar to the one adopted in [9] for clustering large datasets. For each SON, the APS treats its concepts collectively through a summarized representation called *Semantic Feature (SF)*. *SFs* should meet the following requirements:

- incremental updatability whenever a new peer joins the SON;
- speed-up calculation of intra- and inter-cluster measurements;
- adequateness for the computation of distances between SONs and quality metrics of a SON.

Having in mind the above properties, we define the SF_j of a SON SON_j, conceptually represented by the cluster of concepts Cpt_j, as:

- n_j, the Cpt_j cardinality;
- Cl_j, the Cpt_j's clustroid, together with $DistFactor(Cl_j)$ and the unique identifier CP_j of the peer it belongs to (e.g. the IP address);
- r_j, the internal radius which is defined as follows.

  DEFINITION 3 (INTERNAL RADIUS). *Given a set C of n concepts with clustroid Cl, its Internal Radius r is defined as: $r = \sqrt{\frac{DistFactor(Cl)}{n}}$.*

- $samp_j$, a set of p sample concepts selected from Cpt_j, together with their $DistFactor$ values. The number p is appropriately fixed. $samp_j$ is the set of p-nearest concepts from the clustroid Cl_j and have consequently the lowest $DistFactor$ values.

4.2 Exploiting APS for SONs selection

The information stored in the APS provide a summarized description of the existing SONs, $SON_1, \ldots, SON_k$, and thus they are the semantic "beacon" guiding each newly enetring peer P_{new} in selecting its closest SONs.

In particular, when $P_{new} = \{c_1^{new}, ..., c_m^{new}\}$ accesses the APS, it uses clustroids as shown in Algorithm 1 "ChooseSON". The outcome is the placement function ψ_{new} which associates each peer's concept to at most one SON.

The algorithm first computes the matrix $DIST$ of the distances between the m P_{new}'s concepts and the clustroids of the k *SFs*. Starting from the $DIST$ matrix, several criterions could be applied in order to solve the problem of finding the semantically closest SONs for P_{new}. According to classical proposals in the field of clustering (e.g. [9]), the $BestSelection$ function assigns each P_{new}'s concept c_i^{new} to

Algorithm 1 ChooseSON

Require: $P_{new} = \{c_1^{new}, ..., c_m^{new}\}$: joining peer, APS: Access Point Structure, T: threshold
Ensure: ψ_{new} list of selected SONs
1: $DIST_{k \times m}$ matrix of the distances;
2: **for all** c_i^{new} $(i = 1...m)$ **do**
3: **for all** $Cl_j \in SF_j$ $(j = 1...k)$ **do**
4: compute $d(c_i^{new}, Cl_j)$;
5: **if** $d(c_i^{new}, Cl_j) \leq T$ **then**
6: $DIST[j, i] = d(c_i^{new}, Cl_j)$;
7: **else**
8: $DIST[j, i] = \infty$;
9: $\psi_{new} = BestSelection(DIST)$;
10: return ψ_{new};

the SON SON_j for which the distance $d(c_i^{new}, Cl_j)$ from SON_j's clustroid Cl_j is the lowest one.

Notice that in the computation of the $DIST$ matrix a threshold requirement T is used to control the SON tightness or quality. In fact, the distances greater than the threshold T are set to ∞ and ignored in the $BestSelection$ computation. The main idea is that each concept c_i^{new} is inserted into the SON SON_j closest to c_i^{new} if the threshold requirement T is not violated due to the insertion. Otherwise, c_i^{new} forms a new cluster. Therefore, the value of T can be appropriately chosen in order to control the number of SONs and their sizes. In particular, low values of T produce a high number of small and tight SONs. An opposite behavior can be observed with high values of T.

4.3 APS Evolution

The information stored in the APS describes the current setting of the system and should consequently be kept updated to reflect the changes possibly occurring in the network. The evolution of the APS is managed in an incremental fashion as follows.

We first consider an already established network. Each peer P_{new} entering the system executes Algorithm 1 which partially instantiates its placement list ψ_{new} by associating each P_{new}'s concept to the semantically closest SON, if it exists. Then, concepts mapped to the same SON are grouped together. Finally, the peer enters each SON SON_j which is associated to with a non-empty set of concepts, $Cpt_j^{P_{new}}$. Notice that Algorithm 1 works on single concepts, however the monotonicity property of d guaranties that if SON_j is the best choice for each concept in $Cpt_j^{P_{new}}$, then it is the best choice for the whole set $Cpt_j^{P_{new}}$.

This results in SON_j to be conceptually represented by the cluster of concepts $Cpt_j \cup Cpt_j^{new}$ and its SF_j has to be updated accordingly. The number of concepts n_j is augmented and the $DistFactor$ value of each concept in $Cpt_j^{P_{new}}$ must be computed for the incremental maintenance of the p sample concepts and the clustroid. Notice that the exact computation of the above value would require all the SON's concepts, which are not available at the APS level. Nevertheless, an approximate computation for $c \in Cpt_j^{new}$ is possible, which only uses the clustroid Cl_j:

$$\begin{aligned} DistFactor(c) &= \sum_{k=1}^{n_j} d^2(c, c_k) \\ &\approx n_j * r_j^2 + n_j * d^2(c, Cl_j) \end{aligned}$$

The correctness of this heuristic is shown in [9]. Then, also the $DistFactor$ values of the clustroid Cl_j and the p sample concepts are updated under the c insertion. Finally, the concept with the lowest $DistFactor$ value among the new concepts c, the p sample concepts and the old clustroid Cl_j becomes the new clustroid, and the p sample concepts are updated similarly. Finally, the internal radius is updated accordingly.

As to the P_{new}'s concepts for which no corresponding SON has been found, they are clustered in groups and each group forms a new SON. In this case, we adopt a standard agglomerative approach by defining the distance between clusters as an average group linkage [11]. Then a new SF is computed for each newly added SON and inserted in the APS. The above approach is also the one adopted by the first peer entering the system, as no SON exists and algorithm "ChooseSON" returns a void list.

5. CHOOSING NEIGHBORS

When a newly entering peer P_{new} has chosen its semantically closest SONs, it navigates the link structure within each selected SON with the aim of searching for its preferred neighbors, i.e. the semantically nearest peers. In particular, we support two types of neighbor selection: Each peer is allowed to select either the k semantically nearest peers (*k-NN selection*) or the peers in the SON for which the distance between their SON's concepts and the peer SON's concepts are below a given threshold (*range-based selection*). The topology of the network is heavily influenced by the kind of neighbor selection each peer executes when it joins the network. A k-NN selection limits the number of neighbors and thus controls the degree of connectivity. This is not possible in a range-based selection where we can only provide an estimation of the number of neighbors based on the SON statistics we maintain at the APS level.

Adopting a broadcast-based approach to search neighbors could imply wasting precious resources. Indeed, a network of peers is usually required to solve many tasks simultaneously. Instead, we propose that the neighbor selection process be guided by a distributed index mechanism which maintains at each node specifically devised indices named Semantic Clustering Indices (SCIs). The collection of SCIs distributed across the peers P_{new} visits, drives its navigation towards the peers in the same SON containing the concepts nearest to its concepts.

5.1 SCIs

In order to guide a peer joining the network towards its best position in the selected SONs, each SCI maintains summarized information about the SONs'concepts available in each direction. In particular, the SCI, SCI_P, of a peer P is a matrix with $n + 1$ rows and m columns, where n is the number of P's neighbors and m the number of SONs after P joined the network [2]. Each column j contains non-null values in correspondence of each peer belonging to SON_j. More precisely, the first row refers to peer P and thus if P belongs to SON_j then $SCI_P[0, j]$ refers to the set of concepts Cpt_j^P through which P joined SON_j. As to the other rows, if the i-th neighbor is a SON_j's peer then $SCI_P[i, j]$

[2]For ease of explanation, we define SCIs as sparse matrices as each peer usually does not belong to all SONs. Obviously, this does not correspond to their actual implementation.

refers to the set of SON_j's concepts which are reachable in the subnetwork rooted at the i-th neighbor. It indeed corresponds to a sub-SON $SON_{i,j}$ of SON_j [3]. Similarly to the definition of *SF*, each cell $SCI_P[i,j]$ stores the following information about $SON_{i,j}$:

- $Cl_{i,j}$, the clustroid of the sub-SON $SON_{i,j}$, together with $DistFactor(Cl_{i,j})$;
- $r^*_{i,j}$, the *external radius*, i.e. the maximum distance between $Cl_{i,j}$ and the $SON_{i,j}$'s concepts;
- $samp_{i,j}$, a set of $2p$ sample concepts, the p nearest and the p farthest concepts from the clustroid, selected from $SON_{i,j}$ and the corresponding $DistFactor$ values.

One possible configuration of peer Stanford's SCI is shown in Fig. 3.

	SON1	SON2
Stanford	(*papers*,0.6,...)	(*people*,0.3,...)
DBLP	(*article*,0.86,...)	*null*
UParis	*null*	(*members*,0.5,...)
URome	*null*	(*staff*,0.25,...)

Figure 3: The Stanford's SCI

5.2 Using SCIs for Neighbor Selection

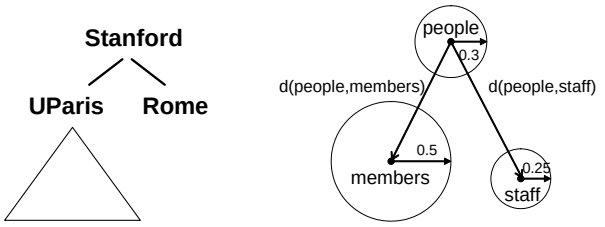
When a newly entering peer joins a SON, the neighbor selection process starts from the clustroid peer of the SON and navigates the SON's link structure by visiting (some of) the peer's immediate neighbors, then their immediate neighbors, and so on, with the aim of finding its semantically nearest peers.

The SCIs distributed across the above mentioned peers provide an abstraction of the network of peers forming the SON as trees. More precisely, if P belongs to SON_j then P is the root node and it has as many children as the number of its neighbors in SON_j, i.e. the number of non-null cells in $SCI_P[*,j]$. The i-th (non-null) neighbor of P roots the sub-SON $SON_{i,j}$ whose properties are described in $SCI_P[i,j]$. In particular, all concepts in $SON_{i,j}$ are within the distance $r^*_{i,j}$ from the clustroid $Cl_{i,j}$. With reference to SON2 in Fig. 3, Fig. 4 depicts this tree-based abstraction from the Stanford point of view. SCIs are used to lighten the neighbor selection process. The objective is to reduce the network load, i.e. the number of accessed peers and the computational effort which is required to each accessed peer. To this end, the information stored in the SCIs are appropriately exploited to effectively apply the triangular inequality to prune out non-relevant peers and to avoid useless distance computations. In this sense, our approach and the algorithm we devised are similar to the work in [4].

5.2.1 Range-based Selection

DEFINITION 4 (RANGE-BASED SELECTION). *Given the set of P_{new}'s concepts $Cpt_j^{P_{new}}$ through which P_{new} joins SON_j and a maximum semantic distance $r(P_{new})$, the range-based selection process $Range(Cpt_j^{P_{new}}, r(P_{new}), SON_j)$ selects all the peers P in SON_j such that $d(Cpt_j^P, Cpt_j^{P_{new}}) \leq r(P_{new})$, where Cpt_j^P is the set of P's concepts associated to SON_j.*

[3] In the following, we will abuse notation by denoting with $SON_{0,j}$ the set of SON_j's concepts in P



(a) Tree abstraction (b) Relationships among concepts

Figure 4: Abstraction of SON2 from peer Stanford point of view

Algorithm 2 RangeSelection

Require: $Cpt_j^{P_{new}}$: concepts, $r(P_{new})$: radius, SON_j: selected SON
1: **if** $d(Cpt_j^P, Cpt_j^{P_{new}}) \leq r(P_{new})$ **then**
2: send(P_{new}, connection request);
3: **for all** $i = 1,\ldots,n$: $SCI_P[i,j] \neq null$ and the i-th neighbor is unvisited **do**
4: **if** $|d(Cpt_j^P, Cpt_j^{P_{new}}) - d(Cl_{i,j}, Cpt_j^P)| \leq r(P_{new}) + r^*_{i,j}$ **then**
5: compute $d(Cl_{i,j}, Cpt_j^{P_{new}})$;
6: **if** $d(Cl_{i,j}, Cpt_j^{P_{new}}) \leq r(P_{new}) + r^*_{i,j}$ **then**
7: send(P_i, $RangeSelection(Cpt_j^{P_{new}}, r(P_{new}), SON_j)$); // P_i is the i-th neighbor

The code each peer executes in order to implement the range-based selection is depicted in Algorithm 2 "RangeSelection". First, it is executed by the clustroid peer Cl_j of the selected SON SON_j. Then, it is recursively executed on all the paths in SON_j originating at Cl_j which cannot be excluded from leading to peers satisfying the range-based selection. Peers interaction is implemented through a message exchange protocol. More precisely, for each contacted peer P, first the condition of the range-based selection is tested (lines 1-2) and, in case, a "connection request" message is sent to P_{new}. Then, the information stored in the column $SCI_P[*,j]$ are used in the distance computation process and the triangular inequality is exploited to avoid the exploration of useless subnetworks (lines 3-7). In particular, if $d(Cl_{i,j}, Cpt_j^{P_{new}}) > r(P_{new}) + r^*_{i,j}$ then, for each concept c in $SON_{i,j}$, $d(c, Cpt_j^{P_{new}}) > r(P_{new})$ (line 7). Moreover, line 4 prunes subnetworks without computing any new distance, since $d(Cl_{i,j}, Cpt_j^P)$ can be computed at compile time and $d(Cpt_j^P, Cpt_j^{P_{new}})$ has already been computed by the calling peer, but the first. The correctness of lines 4 and 6 are shown in [4].

Choosing the range.

One of the main issues related to range-based selection is the choice of the range. Indeed, small ranges run the risk of not finding neighbors whereas large ranges could prove little selective. Furthermore, the selectivity of a range query may vary widely both over time, as the SON evolves, and as a function of location in the SON, which could contain

subgroups that are not well-separated enough to stand as autonomous SONs, although each subgroup contains peers dealing with closely related topics. For this reason, the query range should adapt to the local "density" of the SON. In statistical *nonparametric kernel density estimation* [22], the density at a space point x is computed as a weighted sum of the distances between x and its neighbouring data objects, where weights are monotonically decreasing with distance, i.e., a neighbouring data object exerts more influence on x than any data object that is farther from x. The rate of decrease of the weights is determined by a *kernel function*. Usually, such functions have bounded support or decay very rapidly with distance from x. Thus, in practice only objects in the close neighbourhood of x are involved in the computation of the estimate at x. Using the distance between the sets of representative concepts of peers in kernel estimate, the density value at P_{new} can be used as an adjusting factor for the query range. However, computing the estimate requires a query to determine the peers located in the neighbourhood of P_{new}. An approximate, more efficient approach is, for each SON, to add a description of a small set of *well-scattered* peers in the SON's APS. For each of these peers, its topics (associated to SON_j) and its density estimate values are maintained too. The range of the query is then a decreasing function of the mean density at the nearest m well-scattered peers, where m is a parameter. The clustroid peer updates the values of the density estimates in the APS for all well-scattered peers that are located in the close neighbourhood of P_{new}.

5.2.2 k-NN Selection

DEFINITION 5 (K-NN SELECTION). *Given the set of P_{new}'s concepts $Cpt_j^{P_{new}}$ through which P_{new} joins SON_j and an integer $k \geq 1$, the k-NN selection process $NN(Cpt_j^{P_{new}}, k, SON_j)$ selects the k peers $\{P_1, \ldots, P_k\}$ in SON_j whose concepts associated to SON_j, $Cpt_j^{P_i}$, i=1 ... k, have the shortest distance from $Cpt_j^{P_{new}}$.*

The limit k can be conveniently chosen according to the degree of connectivity we want in the SON and, thus, in the network. As in general, the size of SONs in a network vary widely, it would be inconvenient to set k to a fixed value for all SONs. A starting point to set k could be a slowly increasing sublinear function of the number of peers in the SON the new peer is entering.

For the implementation of the k-NN selection mechanism (see Algorithm 3 "KNNSelection"), we adopt a branch-and-bound technique that is similar to the one devised for the M-Tree [4] and that utilizes two structures: A priority queue PR and a k-elements array NN which, at the end of the execution, contains the results (i.e. the selected neighbors).

PR is a queue of pointers to active subnetworks, i.e. subnetworks where the k nearest neighbors of P_{new} can possibly be found. Each entry of this structure maintains a pointer to the subnetwork (i.e. the peer N from which the subnetwork originates), the clustroid $Cl_{N\bigtriangledown}$ of the subnetwork and a lower bound $d_{min}(N)$ on the distance of any peer in the subnetwork from the joining peer P_{new}:

$$d_{min}(N) = \max\{d(Cl_{N\bigtriangledown}, Cpt_j^{P_{new}}) - r(N\bigtriangledown), 0\}$$

where $r(N\bigtriangledown)$ is the external radius of the subnetwork. This bound is used by the *ChoosePeer* function (line 13) to extract from PR the next subnetwork to be examined in the searching process. The function (not specified here) simply implements an heuristic criterion which selects from PR the node N with the minimum value for $d_{min}(N)$.

As to the NN array, at the end of the execution, the i-th position will contain the value associated to the i-th nearest neighbor of P_{new}, P: $NN[i] = [P, d(Cpt_j^{P_{new}}, Cpt_j^P)]$. If we denote with d_i the distance value in the i-th entry, we have that d_k is the largest distance value in NN and thus d_k can be used as a dynamic search radius, since any subnetwork $N\bigtriangledown$ for which $d_{min}(N) > d_k$ can be safely pruned.

Algorithm 3 KNNSelection

Require: $Cpt_j^{P_{new}}$: concepts, k: integer, SON_j: selected SON

1: **for all** $i = 1, \ldots, n$: $SCI_P[i,j] \neq null$ and the i-th neighbor is unvisited **do**
2: **if** $|d(Cpt_j^P, Cpt_j^{P_{new}}) - d(Cl_{i,j}, Cpt_j^P)| \leq d_k + r^*_{i,j}$ **then**
3: compute $d(Cl_{i,j}, Cpt_j^{P_{new}})$;
4: **if** $d_{min}(P_i) \leq d_k$ **then** // P_i *is the i-th neigh.*
5: add $[P_i, d_{min}(P_i)]$ to PR;
6: **if** $d_{max}(P_i) < d_k$ **then**
7: $d_k = NNUpdate([-, d_{max}(P_i)])$;
8: remove from PR all entries for which $d_{min}(N) > d_k$;
9: **if** $d(Cpt_j^P, Cpt_j^{P_{new}}) \leq d_k$ **then**
10: $d_k = NNUpdate([P, d(Cpt_j^P, Cpt_j^{P_{new}})])$;
11: remove from PR all entries for which $d_{min}(N) > d_k$;
12: **if** $PR \neq \emptyset$ **then**
13: $NextPeer = ChoosePeer(PR)$;
14: send($NextPeer$, $KNNSelection(Cpt_j^{P_{new}}, k, SON_j)$;
15: **else**
16: **for all** $i = 1, \ldots, k$ **do**
17: send(P_i, connect to P_{new}) // P_i *is the peer in the i-th entry of NN*

Initially, the priority queue PR is empty and each entry of NN is set to $[-, \infty]$, i.e. the peers are undefined and the distances are set to ∞. Algorithm 3 is first executed by the clustroid peer CP_j of the selected SON SON_j. Then, it is recursively executed by each peer the *ChoosePeer* function extracts from PR (lines 13-14).

For each contacted peer, the algorithm first determines the active subnetworks and inserts them into the PR queue (lines 1-5). Notice that the same pruning condition as for the range-based selection is exploited at line 2, where the dynamic search radius d_k is used instead of the search radius. Then, if needed, it calls the $NNUpdate$ function to perform an ordered insertion in the NN array and receives back a (possibly new) value of d_k (lines 6-7 and line 9-10). The newly computed d_k values are then used in lines 8 and 11 to remove from PR all subnetworks for which the d_{min} lower bound exceeds d_k. In particular, for lines 6-7 the idea is to compute an upper bound $d_{max}(P_i)$ on the distance of any peer in the P_i's subnetwork from P_{new}: $d_{max}(N) = d(Cl_{N\bigtriangledown}, Cpt_j^{P_{new}}) + r(N\bigtriangledown)$. Then, line 7 inserts $d_{max}(P_i)$ in appropriate position in the NN-array, just leaving the other field undefined. Lines 9-10, instead, checks whether to add the current peer P in the NN array.

At the end of the process, NN contains the k-nearest neighbors of P_{new} and the connections are created (lines 16-17).

5.3 SCIs Construction and Evolution

Since SCIs maintain summarized information about SONs, they change whenever the SONs themselves are modified by the joining or leaving of peers. In particular, SCIs creation and evolution is managed in an incremental fashion as follows.

As a base case, the SCI of an isolated peer P has a single row, referring to the peer P itself. This row expresses the information about the local knowledge of P and contains in each cell $SCI_P[0, j]$ the description of the SON_j which only contains peer P. The clustroid and the $2p$ sample objects are selected from set of concepts Cpt_j^P which has been determined when accessing the APS (see Sec. 4.3) and the external radius is computed accordingly.

When an entering peer P_i joins SON_j with its concepts $Cpt_j^{P_i}$ and selects P_k with its concept $Cpt_j^{P_k}$ as its neighbor, a new connection is created.

In particular, each of the connecting peers (say P_i) informs the other peer (say P_k) of the sub-SON that can be accessed through it. To this end, P_i *aggregates* its own SCI by rows and sends the j-th column to P_k. It represents the sub-SON $SON_{i,j}$ and it is obtained by merging the clusters of concepts represented in each cell of the j-th column. The merge of two clusters represented as $(Cl_s, r_s^*, samp_s)$ and $(Cl_t, r_t^*, samp_t)$ is the cluster $(Cl_u, r_u^*, samp_u)$ obtained as follows. The clustroid Cl_u and the sample peer $samp_u$ are chosen among the clustroids, Cl_s and Cl_t, and sample peers, $samp_s$ and $samp_t$, on the basis of the $DistFactor$ values which are computed on the above set of concepts. The external radius is computed according to the definition given in section 5.1 and on this set of concepts too.

After P_k receives such knowledge, it puts the received information in the cell $SCI_{P_k}[i, j]$ (i.e. it adds peer P_i to its SON_j's neighbors), after having extended its SCI with a new row for P_i, if P_i is a newly added neighbor.

Afterwards, both peers P_i and P_k need to inform their own reverse neighbors that a change occurred in the network and thus they have to update their SCIs accordingly. To this end, each peer, say P_i, sends to each reverse neighbor P_h which belongs to the SON SON_j, i.e. for which $SCI_{P_i}[h, j]$ is not empty, an aggregate of its j-th column excluding the P_h's cell. When P_h receives such aggregated information, it updates the cell $SCI_{P_h}[i, j]$ with the received information.

Disconnections are treated in a similar way as connections. When a node disconnects from the network, each of its neighbors must delete the row of the disconnected peer from its own SCI and then informs the remaining neighbors that a change on its own subnetwork has occurred by sending new aggregates of its SCI to them.

6. QUERY PROCESSING

Most works on SONs mainly focus on how query processing in a P2P network can be made more efficient in a well-arranged network (see Sec. 8 for related works). Indeed, a well-clustered network undoubtedly influences positively the query answering process: The search would primarily be directed towards peers belonging to the SONs of interest, thus reducing the number of contacted peers in the network, as well as increasing the chance of finding relevant results.

It is important to emphasize that this paper is more properly concerned with the *efficient construction* and *maintenance* of well-clustered SONs with the aim of reducing semantic degradation during query processing. In a PDMS, a query is formulated over a peer's schema, the peer's mappings are used to reformulate the query over its immediate neighbors and the same recursively applies to each receiving peer. Therefore the ability to obtain relevant data from other nodes depends on the existence of a semantic path of mappings to that node. In this context, semantic degradation is due to the information loss along long paths, because of missing or incomplete mappings, and can even imply the loss of relevant nodes at all. In a PDMS which adopts the network organization we propose instead of a random one, it is more likely that closely associated peers are relevant to the same queries thus reducing semantic degradation. The main idea is that each query concept will be "solved" by the SON which is responsible for it. To this end, besides mappings, also the SCI could be exploited to select the right neighbors, i.e. those ones belonging to the involved SONs.

Query execution can be further improved by query routing, i.e. the process of selecting a small subset of relevant peers to forward a query to. This is a crucial issue, since approaches which flood the SON with lots of messages are not adequate, primarily for efficiency reasons. For this purpose, the semantic infrastructure we presented in this paper can be conveniently complemented with our previous work on Semantic Routing Indices (SRIs) [17], a fully distributed indexing mechanism which summarizes the semantics underlying whole subnetworks. SRIs are exploited to select the best directions to which forward each query to, according to the estimated semantic degradation of information they maintain [18]. As a proof of the applicability of our proposal, we employed the SRIs in the simulations we conducted in the experimental evaluation described in the following section.

7. EXPERIMENTS

In this section we discuss a selection of the experiments we performed to test the effectiveness and the efficiency of the presented approach.

7.1 Experimental Setting

For our experiments we used the SUNRISE simulation framework [19] through which we were able to reproduce the main conditions characterizing a PDMS environment. Through this framework we modelled and generated scenarios corresponding to networks of semantic peers, each with its own schema describing a particular reality. The schemas are derived from real-world data sets, collected from many different available web sites, such as the DBLP Computer Society Bibliography and the ACM SIGMOD Record, and enlarged with new schemas created by introducing variations on the original ones. Among the collections we employed to test our approach, we selected two of them, Collection1 and Collection2, differing for their grade of semantic heterogeneity: each schema in Collection2 covers a higher number of topics than a schema in Collection1, resulting in a more complex scenario. The representation of each peer is derived from its schema by following an approach similar to [26]. For both collections, each representation contains about a dozen of concepts, while the collection's size is in the order of some hundreds of schemas. Further, in order to avoid the presence of cyclic paths in the SCI updates propagation, a cycle detection mechanism based on global unique identifiers is adopted.

7.2 Effectiveness

In the following we demonstrate the effectiveness of our approach from two different points of view: Firstly, we measure the quality of the generated SONs, considered as clusters of concepts, by means of well known quality metrics. Then, we evaluate the quality of the resulting network both by means of theoretical quality indices and by executing complex query processing simulations.

7.2.1 Clustering Quality Indices

In order to assess the quality of the generated SONs on the basis of well-known data clustering quality indices, we first considered SONs as simple clusters of concepts. In particular, we evaluated internal, i.e. intrinsic, quality by means of the Silhouette indices [24] and external, i.e. w.r.t. an ideal clustering, quality by means of the Rand Index [23]. As to internal quality, two were the aspects which interested us: The homogeneity of each cluster and the separation between the obtained clusters (the higher are their homogeneity and separation the better is the clustering). Figure 5-a and 5-b show the different Silhouette values (a,b and s) for Collection1 and Collection2, respectively, using the GM distance and for different threshold values. While Silhouette a factor captures homogeneity by means of a value between 0 (complete homogeneity) and 1 (null homogeneity), Silhouette b factor indicates separation (0 for null separation, 1 for complete separation). Finally, Silhouette s factor captures both aspects in a global internal quality value between -1 (bad clustering) and 1 (very good clustering). As can be seen at first glance from the two figures, the clustering quality offered by our approach is very high for both collections, with low a and high b and s values. More specifically, Figure 5-a shows very good clustering performance for central values of T (from 0.3 to 0.7); for values of T which are too low (e.g. 0.1) cluster separation drops, while for $T \geq 0.9$ cluster homogeneity is no longer optimal. Figure 5-b confirms the goodness of the resulting clustering also for the more complex Collection2, in particular for central values of T.

Since clustering quality is heavily affected by the choice of the distance d, we also tested it for the LC distance: Figure 6-a shows the obtained results for Collection2 (Collection 1 produced a similar trend and will not be presented). Again, the clustering quality is very high; in this case, values of T between 0.4 and 0.6 appear as the best choice, since clustering homogeneity and separation are still high (low a and high b, respectively).

In order to further evaluate clustering quality, we also analyzed external quality by computing the Rand Index between the obtained clustering and an ideal clustering situation, where we manually created clusters of related concepts. Rand Index is a measure of similarity ranging from 0 to 1 and since we compute it w.r.t. an ideal situation, values near 1 will denote optimal clusterings. Figure 6-b depicts the results for different values of T and for the same collections and distances we previously considered for the internal quality tests. The very high values we obtain (near 1 for Collection2, 1 for Collection1) denote very good clustering effectiveness. In particular, both collections achieve optimal clustering with the GM distance and for values of T between 0.3 and 0.7. On the other hand, confirming the internal quality results and considerations, the LC distance seems to benefit from lower T values (between 0.2 and 0.4), producing nonetheless very satisfying results in this range (for instance Rand Index = 0.913 for $T = 0.3$). Since, as we have shown, both distances deliver very good and similar results, for the following tests we will consider only the GM one; however, all the considerations that will be made are general and also apply to the LC one.

7.2.2 Network Quality Index

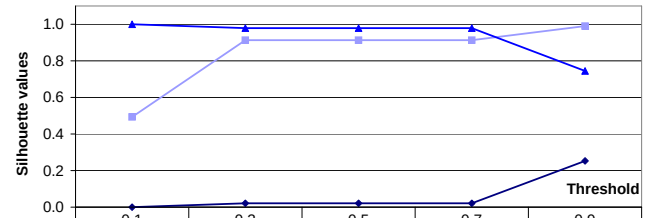
Similarly to the Rand Index tests for clustering quality, in order to perform a preliminary theoretical quality evaluation of the resulting link structures, we manually designed ideal networks of peers considering small subsets (some dozens of schema descriptions) of the collections. Those networks are ideal from a semantic point of view, i.e. they connect all and only the peers with highly related concepts and topics. Then, we computed the similarity between the automatically generated topologies and the ideal ones, by means of a specifically devised index we called NetIndex. The index is computed as follows: For each direct connection between two peers in the ideal network, the shortest path between them in the real network is found (in the optimal case this would be 1 step long, i.e. no additional semantic approximation is introduced); then the inverse of the mean of these shortest paths lengths is calculated. Thus, an index of 1 denotes a semantically ideal network configuration.

We computed NetIndex for both collections; Figure 7 shows the obtained results for the most complex Collection2 and for both Range (Figure 7-a) and k-NN (Figure 7-b) selections and for different values of threshold T. Let us start by considering, for instance, a typical value for T, $T = 0.5$. As we expected, the NetIndex trends are growing for increasing values of radius (range) and k (k-NN), since the semantically ideal networks are very complex and thus the high number of connections is better approximated by larger radiuses and ks. However, as we can see, a radius of 0.2 or a k of 6, while avoiding to produce over-connected and, thus, possibly inefficient networks, already achieve very good semantic optimality grades (NetIndex values of 0.97 and 0.83, respectively). Finally, different thresholds such as $T = 0.3$ and $T = 0.7$ produce equally good results, while too high ($T = 0.9$) or too low ones ($T = 0.1$) produce semantically inferior network configurations.

7.2.3 Network Quality for Query Processing

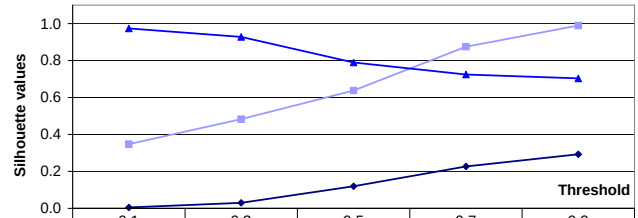
In order to evaluate the impact of the network organization we propose on query answering, we simulated a querying process on the networks produced by our algorithms. The results we present here refer to the execution of our k-NN selection algorithm on both collections. In particular, we choose a value of k=6, since, as shown by the previous experiments, it allows us to have networks with a high NetIndex value, while maintaining the low grade of connectivity required in a realistic context. However, the results we show here are indeed representative of the entire set of obtainable networks organizations.

As to the querying process, we simulated it by instantiating different queries on randomly selected peers where each query is a combination, through logical connectives, of a small number of predicates specifying conditions on concepts. More precisely, we quantified the advantages on query processing by propagating each query until a specific stopping condition is reached [18]: We evaluated the effectiveness improvement by measuring the quality of the results (*satisfaction*) when a given number of hops (*hop*) has been



	0.1	0.3	0.5	0.7	0.9
Sil. a	0.0000	0.0212	0.0212	0.0212	0.2528
Sil. b	0.4931	0.9133	0.9133	0.9133	0.9900
Sil. s	1.0000	0.9786	0.9786	0.9786	0.7446

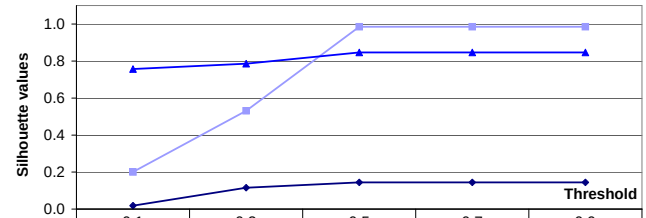
(a) Internal quality - Collection1 - GM distance



	0.1	0.3	0.5	0.7	0.9
Sil. a	0.0047	0.0301	0.1196	0.2264	0.2930
Sil. b	0.3475	0.4827	0.6380	0.8752	0.9894
Sil. s	0.9738	0.9285	0.7895	0.7241	0.7036

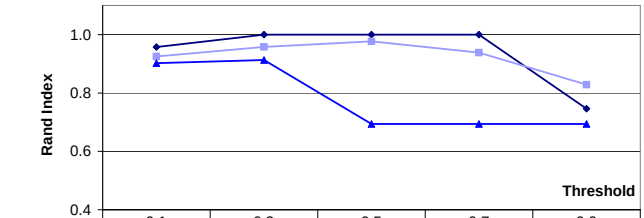
(b) Internal quality - Collection2 - GM distance

Figure 5: Clustering quality: Internal indices



	0.1	0.3	0.5	0.7	0.9
Sil. a	0.0187	0.1157	0.1446	0.1446	0.1446
Sil. b	0.2003	0.5311	0.9849	0.9849	0.9849
Sil. s	0.7564	0.7854	0.8467	0.8467	0.8467

(a) Internal quality - Collection2 - LC Distance



	0.1	0.3	0.5	0.7	0.9
Coll1, GM	0.9577	1.0000	1.0000	1.0000	0.7460
Coll2, GM	0.9253	0.9581	0.9771	0.9383	0.8287
Coll2, LC	0.9022	0.9131	0.6940	0.6940	0.6940

(b) External quality

Figure 6: Additional clustering quality tests

performed or, in a dual way, the efficiency improvement by measuring the number of hops required to reach a given satisfaction goal. Satisfaction is a specifically introduced quantity that grows proportionally to the goodness of the results returned by each queried peer. The search strategy employed is the depth-first search (DFS).

For the selection of the most promising neighbor, besides the simplest random strategy (*Rnd*), *SRI* [17] is experienced too (see Sec. 6). For both these routing strategies, we compared the results measured in the clustered network (*Cls*) generated with our k-NN selection process with a random one (*Rnd*). Notice that all the results we present are computed as a mean on several hundreds query executions.

Figures 8-a and 8-b show the trend of the obtained satisfaction when we gradually vary the stopping condition on hops for Collection1 and Collection2, respectively. As we expected, in Figure 8-a we can see that both for the random and the SRI routing strategies, the obtained values of satisfaction are higher in the clustered network. Further, notice that even in Figure 8-b, which refers to the very complex scenario represented by Collection2, the clustered network shows a better behaviour.

Figure 9 represents the dual situation, where the number of hops required to reach a given satisfaction goal is measured. For both these graphs we can see that the curves associated to the clustered network outperform in efficiency the corresponding ones for the un-clustered situation. For example, for a satisfaction goal of 4, we obtained 35% of saved hops for the random routing strategy and even 50% of saved hops for the SRI one.

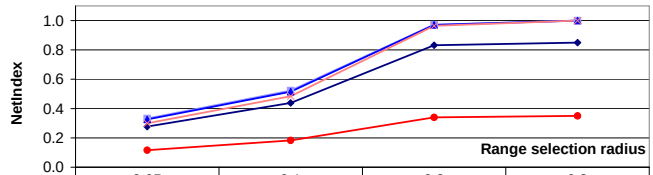
Figures 10-a and 10-b show another measure of effectiveness, which quantifies the percentage of satisfied queries (i.e. the queries for which the given satisfaction goal is reached) for different satisfaction level. Clearly, for all the depicted curves, the percentage of satisfied queries decreases for increasing values of satisfaction, since they are more difficult to reach. Nevertheless, the curves corresponding to the clustering networks always outperform the random ones.

7.3 Efficiency

In order to evaluate the efficiency of our approach we firstly considered the CPU load generated on each peer by the execution of our algorithms of neighbors selection. In particular, we wanted to quantify the percentage (*savedCPU*) of useless distance computations that we are able to avoid thanks to the exploitation of the triangular inequality (see Section 5.1).

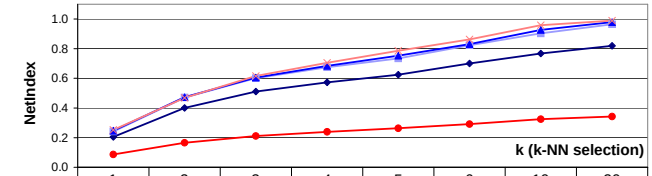
Then, we focused on the number of peers that is necessary to contact in order to solve our neighbors selection processes. In particular, since the solution of a naive k-NN or range selection would require the exploration of all the peers in the considered SON, we were interested in quantifying the percentage (*savedPeers*) of useless peers that we can prune out thanks to our algorithms.

Figures 11-a and 11-b show the savedCPU and savedPeers values obtained for Collection2 executing range and k-NN selection processes, respectively. The results are collected for different values of radius and k. As we can see, for both graphs the obtained results are high, signifying a great saving for the CPU load and the number of contacted peers. For example, for a range selection process with a radius of 0.05 we have 48.75% of savedCPU and 77.39% of saved-



	0.05	0.1	0.2	0.3
T=0.1	0.2774	0.4384	0.8320	0.8500
T=0.3	0.3325	0.5235	0.9720	1.0000
T=0.5	0.3263	0.5158	0.9700	1.0000
T=0.7	0.3000	0.4842	0.9640	1.0000
T=0.9	0.1164	0.1832	0.3402	0.3500

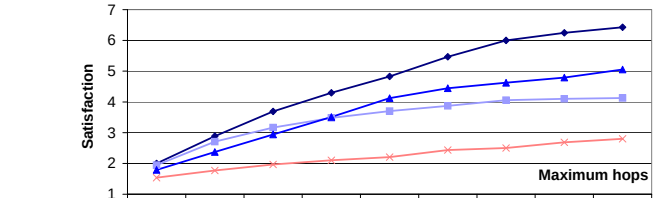
(a) Range selection - Collection2



	1	2	3	4	5	6	10	20
T=0.1	0.2051	0.4006	0.5111	0.5734	0.6243	0.7000	0.7679	0.8199
T=0.3	0.2413	0.4713	0.6013	0.6746	0.7345	0.8235	0.9034	0.9645
T=0.5	0.2474	0.4737	0.6053	0.6842	0.7532	0.8316	0.9263	0.9789
T=0.7	0.2526	0.4684	0.6158	0.7053	0.7865	0.8632	0.9579	0.9895
T=0.9	0.0866	0.1658	0.2118	0.2395	0.2636	0.2911	0.3242	0.3426

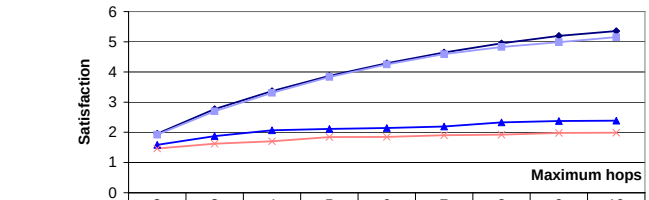
(b) k-NN selection - Collection2

Figure 7: NetIndex network quality



	2	3	4	5	6	7	8	9	10
Cls/SRI rout.	2.0000	2.8878	3.6927	4.2997	4.8267	5.4670	5.9989	6.2454	6.4273
Rnd/SRI rout.	1.9429	2.7100	3.1637	3.4787	3.7008	3.8711	4.0556	4.1001	4.1291
Cls/rnd rout.	1.7889	2.3731	2.9431	3.5099	4.1256	4.4443	4.6244	4.7914	5.0540
Rnd/rnd rout.	1.5365	1.7714	1.9673	2.1023	2.2063	2.4370	2.5011	2.6861	2.8036

(a) Collection1 (k-NN network, k=6)



	2	3	4	5	6	7	8	9	10
Cls/SRI rout.	1.9502	2.7686	3.3683	3.8712	4.2881	4.6463	4.9490	5.1944	5.3516
Rnd/SRI rout.	1.9214	2.6959	3.3082	3.8319	4.2481	4.5906	4.8290	4.9885	5.1524
Cls/rnd rout.	1.5848	1.8753	2.0675	2.1106	2.1420	2.1912	2.3297	2.3734	2.3865
Rnd/rnd rout.	1.4636	1.6256	1.7056	1.8468	1.8509	1.9012	1.9198	1.9772	1.9843

(b) Collection2 (k-NN network, k=6)

Figure 8: Network quality for query processing: Reached satisfaction for a given number of hops

Peers, while for a k-NN selection with k=4 we have 49.24% of savedCPU and 25% of savedPeers. Finally, notice that the savedPeers factor decreases with the radius and k values, since augmenting them results in a higher number of peers to connect to and, consequently, to contact (while the number of peers in the SONs remains unchanged).

8. RELATED WORK AND CONCLUSIONS

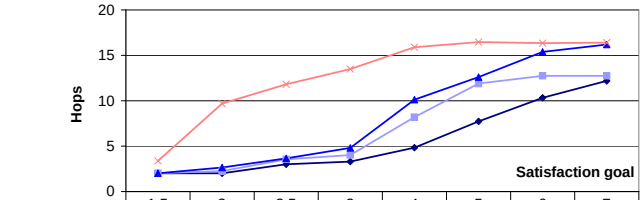
PDMSs [10] naturally support the autonomy of peers in the network, allowing for a personalized semantic marking up of data to cohabit nicely with a collaborative exchange of data, which is possible thanks to the use of semantic mappings established between the peers. The work presented in this paper relies on a PDMS as a reference scenario. One crucial issue in a PDMS is about query processing which may lead to very poor results. In such a context, the neighborhood of semantically similar peers is of decisive importance.

In P2P systems, being able to improve the retrieval of data while maintaining a high degree of peers' autonomy is exactly the goal pursued by one of the most recent evolution in network organization strategies: Semantic Overlay Networks (SONs). The key idea is to exploit self-organization principles in order to "cluster" together semantically similar peers and, thus, similar contents. The original SON idea first appeared in [6], where peers are assigned to SONs according to classification hierarchies which are defined a-priori and shared in the network. In a PDMS scenario this approach is not applicable because of autonomy of peers in defining their schemas and the absence of a common agreement as to the vocabularies used at the peers' schemas. A further difference with our proposal is that classification hierarchies maintain a direct reference to each peer belonging to the overlay networks, thus making the approach not much scalable. Finally, [6] ignores the link structure within the overlay network and a SON is represented just by a set of peers.

Other works address the problem of building SONs. In the following, we will discuss those ones sharing some aspects with our approach.

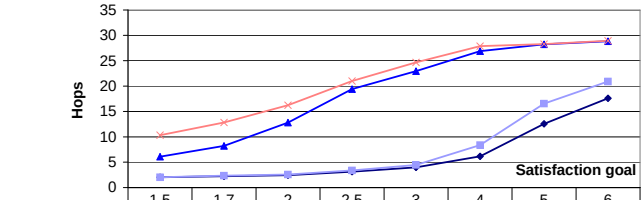
The papers [1, 5, 15] adopt gossip-based membership protocols to derive neighborhoods and, then, SONs translate to the sets of those peers which are logically interconnected. In the schema-based P2P scenario considered in [1], semantic gossiping is applied to derive semantic neighborhoods, i.e. sets of peers that have annotated their data according to the same schema. This approach imposes the peers in the network to conform their data to local schemas derived by inheritance from a set of base schemas. A similar approach is also adopted in [5] where the resulting unstructured network coexists with a DHT-based structured one in an hybrid topology. In [15], a metric distance for Language Models is adopted to compare the peers' local collections of documents. Whenever a meeting occurs for a k-NN selection, the two peers decide whether they should become neighbors. Then they exploit the triangular inequality property to compute a lower-bound on the distance between each peer and the other peer's neighbors.

Clustering principles similar to the ones presented in this paper are explored in [16] with application to Web-scale data integration. However, only a general discussion is provided and no details are given about the underlying technical challenges. Also the works in [2, 7, 14, 25] found on principles of clustering but with the aim of providing answers to the



	1.5	2	2.5	3	4	5	6	7
Cls/SRI rout.	2.0000	2.0000	3.0000	3.2929	4.82828	7.7253	10.3299	12.1818
Rnd/SRI rout.	2.0202	2.2525	3.5455	4.0202	8.18182	11.8889	12.7374	12.7374
Cls/rnd rout.	2.0000	2.6465	3.6566	4.8085	10.1111	12.5979	15.3737	16.1919
Rnd/rnd rout.	3.3721	9.6837	11.8081	13.4848	15.8889	16.4545	16.3434	16.4040

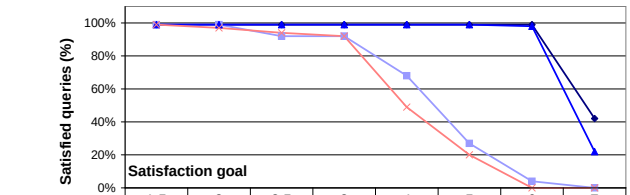
(a) Collection1 (k-NN network, k=6)



	1.5	1.7	2	2.5	3	4	5	6
Cls/SRI rout.	2.0100	2.2234	2.4343	3.1212	3.9697	6.1538	12.5408	17.5960
Rnd/SRI rout.	2.0210	2.3422	2.5657	3.3636	4.4545	8.3838	16.5455	20.8990
Cls/rnd rout.	6.0909	8.2188	12.7980	19.4184	22.9495	26.8980	28.2424	28.8443
Rnd/rnd rout.	10.3232	12.8081	16.2323	20.9899	24.6633	27.8788	28.2828	28.9435

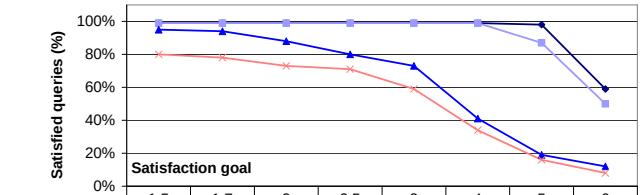
(b) Collection2 (k-NN network, k=6)

Figure 9: Network quality for query processing: Required number of hops for a given satisfaction goal



	1.5	2	2.5	3	4	5	6	7
Cls/SRI rout.	99%	99%	99%	99%	99%	99%	99%	42%
Rnd/SRI rout.	99%	99%	92%	92%	68%	27%	4%	0%
Cls/rnd rout.	99%	99%	99%	99%	99%	99%	98%	22%
Rnd/rnd rout.	99%	97%	94%	92%	49%	20%	0%	0%

(a) Collection1 (k-NN network, k=6)



	1.5	1.7	2	2.5	3	4	5	6
Cls/SRI rout.	99%	99%	99%	99%	99%	99%	98%	59%
Rnd/SRI rout.	99%	99%	99%	99%	99%	99%	87%	50%
Cls/rnd rout.	95%	94%	88%	80%	73%	41%	19%	12%
Rnd/rnd rout.	80%	78%	73%	71%	59%	34%	16%	8%

(b) Collection2 (k-NN network, k=6)

Figure 10: Network quality for query processing: % of satisfied queries for a given satisfaction goal

issue of how to actually create SONs of autonomous peers for efficient IR search. In [7] the clustering method is guided by description of peers in terms of processing power, storage capacity, etc., and seems not to exploit semantic similarity criteria on the peers' contents. Then, the network of peers results in a hierarchical organization. Instead, [2, 14] propose to cluster together peers sharing objects with similar representations and maintains intra- and inter-cluster connections. However, both approaches do not support multiple classifications and policies to select neighbors. Each peer within a cluster chooses neighbors at random [2] or knows each other [14]. Moreover, the clustering approach proposed in [2] is not incremental and clustroids are recomputed at regular intervals of time. In [25], the main focus is to ensure load balancing in a network where nodes are logically organized into a set of clusters and where a set of super-peer nodes maintains information about which documents are stored by which cluster nodes.

Hierarchical organization are also adopted in [20, 12] with the aim of providing an efficient query processing support in schema-based P2P networks. In both approaches a set of designed nodes (named super-peers in two-level hierarchies [20] or root-nodes in multi-level hierarchies [12]) are connected to a main channel that provides communication among them. Then, peers with similar characteristics populate the same hierarchy. Such kinds of network organization are too restrictive for our connection purposes.

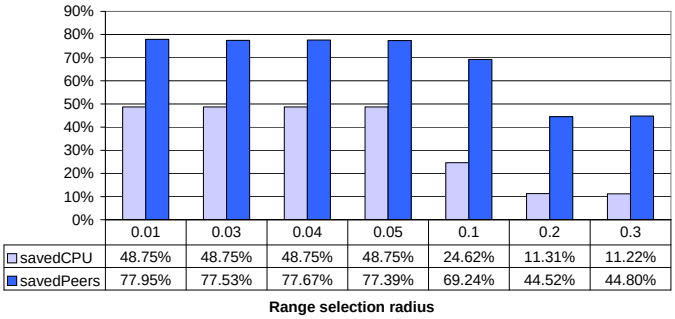
In [21] a different perspective is considered. The approach relies on caching high quality peers to be remembered as "friends" to be connected to. However, friend lists have a fixed length, thus imposing to the peers a predefined maximum number of neighbors. Then, semantic similarity between peers is based on term frequency distributions, thus implicitly assuming a common vocabulary in the network. A further difference with [21] is that in our proposal the choice of the neighbors is *efficiently* supported by specifically devised algorithms which exploit a semantic-based distributed indexing mechanism.

Our approach takes up the clustering purpose originally proposed in [6] by effectively and efficiently supporting the peers to locate their semantically best connections in a SON. As to the process of creation and maintenance of a SON, we inspired to the BUBBLE framework [9], a seminal work in the field of incremental clustering of large datasets in metric spaces. Indeed, BUBBLE stands out as one of the very few approaches offering incrementality, scalability and small memory requirements, and the ability to cluster objects described by heterogeneous vocabularies, since no assumptions are made on the distance function, and the distance of any concept pair is user-definable. For what concerns the indexing mechanism used when navigating the SON in search of the semantically best neighbors, the spirit of our work is similar to the proposal for metric spaces presented in [4].

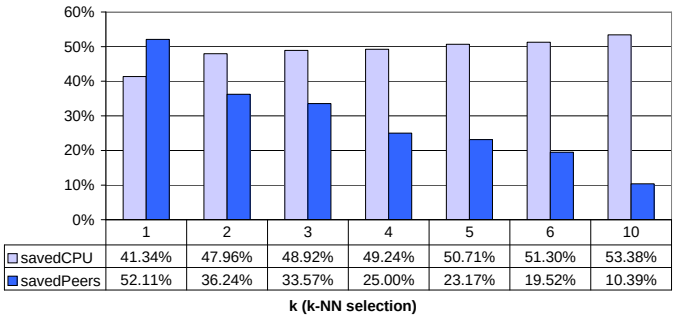
In our future work, we plan to enhance the proposed approach by adding merging and splitting in the SON management and by deepening the problem of range and k selection. Moreover, we will investigate strategies for query processing in this context.

9. REFERENCES

[1] K. Aberer, P. Cudré-Mauroux, M. Hauswirth, and T. V. Pelt. GridVine: Building Internet-Scale



(a) Range selection - Collection2



(b) k-NN selection - Collection2

Figure 11: Efficiency tests: % of saved computations and contacted peers

Semantic Overlay Networks. In *Proc. of ISWC*, pages 107–121, 2004.

[2] M. Bawa, G. Manku, and P. Raghavan. SETS: Search Enhanced by Topic Segmentation. In *Proc. of the 26th ACM SIGIR Conf.*, pages 306–313, 2003.

[3] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, May 2001.

[4] P. Ciaccia, M. Patella, and P. Zezula. M-tree: An Efficient Access Method for Similarity Search in Metric Spaces. In *Proc. of the 23rd VLDB Conf.*, pages 426–435, 1997.

[5] C. Comito, S. Patarin, and D. Talia. PARIS: A Peer-to-Peer Architecture for Large-Scale Semantic Data Integration. In *Proc. of the DBISP2P Workshop*, pages 163–170, 2005.

[6] A. Crespo and H. Garcia-Molina. Semantic Overlay Networks for P2P Systems. In *Proc. of the 3rd AP2PC Workshop*, pages 1–13, 2004.

[7] C. Doulkeridis, K. Nørvåg, and M. Vazirgiannis. DESENT: Decentralized and Distributed Semantic Overlay Generation in P2P Networks. *IEEE J. on Selected Areas in Comm.*, 25(1):25–34, 2007.

[8] P. Ganesan, H. Garcia-Molina, and J. Widom. Exploiting Hierarchical Domain Structure to Compute Similarity. *ACM TOIS*, 21(1):64–93, 2003.

[9] V. Ganti, R. Ramakrishnan, J. Gehrke, A. Powell, and J. French. Clustering Large Datasets in Arbitrary Metric Spaces. In *Proc. of the 15th ICDE Conf.*, pages 502–511, 1999.

[10] A. Halevy, Z. Ives, J. Madhavan, P. Mork, D. Suciu, and I. Tatarinov. The Piazza Peer Data Management System. *IEEE TKDE*, 16(7):787–798, 2004.

[11] A. Jain, M. Murty, and P. Flynn. Data Clustering: A Review. *ACM Comp. Surv.*, 31(3):264–323, 1999.

[12] G. Koloniari and E. Pitoura. Content-Based Routing of Path Queries in Peer-to-Peer Systems. In *Proc. of the 9th EDBT Conf.*, pages 29–47, 2004.

[13] C. Leacock and M. Chodorow. Combining Local Context and WordNet Similarity for Word Sense Identification. In C. Fellbaum, editor, *WordNet: An Electronic Lexical Database*, pages 256–283. MIT Press, 1998.

[14] M. Li, W.-C. Lee, and A. Sivasubramaniam. Semantic Small World: An Overlay Network for Peer-to-Peer Search. In *Proc. of the 12th IEEE ICNP*, pages 228–238, 2004.

[15] A. Linari and G. Weikum. Efficient Peer-to-Peer Semantic Overlay Networks Based on Statistical Language Models. In *Proc. of the P2PIR Workshop (in conj. with CIKM)*, pages 9–16, 2006.

[16] J. Madhavan, S. Cohen, X. Dong, A. Halevy, S. Jeffery, D. Ko, and C. Yu. Web-Scale Data Integration: You Can Afford to Pay as You Go. In *CIDR*, pages 342–350, 2007.

[17] F. Mandreoli, R. Martoglia, W. Penzo, and S. Sassatelli. SRI: Exploiting Semantic Information for Effective Query Routing in a PDMS. In *Proc. of the WIDM (in conj. with CIKM)*, pages 19–26, 2006.

[18] F. Mandreoli, R. Martoglia, W. Penzo, S. Sassatelli, and G. Villani. SRI@work: Efficient and Effective Routing Strategies in a PDMS. In *In Proc. of the 8th WISE Conf.*, pages 285–297, 2007.

[19] F. Mandreoli, R. Martoglia, W. Penzo, S. Sassatelli, and G. Villani. SUNRISE: Exploring PDMS Networks with Semantic Routing Indexes. In *Proc. of ESWC*, 2007.

[20] W. Nejdl, M. Wolpers, W. Siberski, C. Schmitz, M. Schlosser, I. Brunkhorst, and A. Löser. Super-Peer-Based Routing and Clustering Strategies for RDF-Based Peer-to-Peer Networks. In *Proc. of the 12th WWW Conf.*, pages 536–543, 2003.

[21] J. Parreira, S. Michel, and G. Weikum. P2PDating: Real Life Inspired Semantic Overlay Networks for Web Search. *Inf. Proc. & Manag.*, 43(3):643–664, 2007.

[22] E. Parzen. On Estimation of a Probability Density Function and Mode. *Ann. Math. Statist.*, 33:1065–1076, 1962.

[23] W. M. Rand. Objective Criteria for the Evaluation of Clustering Methods. *J. Amer. Stat. Assoc.*, 66(336):846–850, 1971.

[24] P. Rousseeuw. Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis. *J. Comput. Appl. Math.*, 20(1):53–65, 1987.

[25] P. Triantafillou, C. Xiruhaki, M. Koubarakis, and N. Ntarmos. Towards High Performance Peer-to-Peer Content and Resource Sharing Systems. In *Proc. of the 1st CIDR*, 2003.

[26] C. Yu and H. Jagadish. Schema Summarization. In *Proc. of the 32nd VLDB Conf.*, pages 319–330, 2006.