NeP4B
Networked Peers for Business

WP4
Task T4.3
Deliverable D4.3.1

# *Design of procedures and structures for the support of the inter-peer semantic queries*

(FINAL, 20/05/2009)

**Abstract –**
In a P2P distributed scenario as the one of the NeP4B, where query answers can come from any peer connected through a semantic path a key challenge is the *query routing*. The routing is the capability of selecting a small subset of relevant peers to forward a query to. Flooding-based techniques are indeed not adequate for both efficiency and effectiveness reasons: not only they overload the network (forwarded messages and computational effort required to solve queries), but also overwhelm users with a large number of results, mostly irrelevant.
In this deliverable we present an innovative mechanism for a unified data retrieval for both semantic and multimedia content.

Claudio Gennaro
ISTI-CNR

# Document information

| | | | |
|---|---|---|---|
| Document ID code | D4.3.1 | | |
| Keywords | semantic routing index, multimedia routing index, routing policies, query rewring | | |
| Classification | **FINAL** | Date of reference | **20/05/2009** |
| Distribution level | **NeP4B Consortium** | | |

| | | |
|---|---|---|
| Editor | Claudio Gennaro | ISTI-CNR |
| Authors | Claudio Gennaro | ISTI-CNR |
| | Federica Mandreoli | Unimo |
| | Riccardo Martoglia | Unimo |
| | Matteo Mordacchini | ISTI-CNR |
| | Rita Lenzi | Unimo |

| Version history | | |
|---|---|---|
| *Date* | *Version* | *Description* |
| 19/02/2009 | DRAFT | Initial version |
| 14/05/2009 | DRAFT | Draft version |

# Contents

# 1 Executive summary

This report presents the activities conducted within task T4.3 of the NeP4B project. It describes two routing technics based on semantic and multimedia query routing and proposes to combine these two approaches in order to design an innovative mechanism which exploits the two main aspects characterizing the querying process in such a context: the semantics of the concepts in the peers' ontologies and the multimedia contents in the peers' repositories.

In this context, we pursue *effectiveness* by selecting, for each query, the peers which are semantically best suited for answering it, i.e. whose answers best fit the query conditions. For what concerns to the execution of multimedia predicates, which is inherently costly we instead pursue *efficiency* by limiting the forwarding of a query towards the network's zones where potentially matching instances are more likely to be found, while pruning the others. In this way, we give the user a higher chance of receiving first the answers which better satisfy the query conditions.

This report provides a formal settlement for the integration of query routing techniques for both semantic and multimedia data which is founded on a fuzzy framework.

This work is the product of the collaboration of the IsGroup of University of Modena and Reggio Emilia and the research unit of the ISTI-CNR.

# 2 Introduction

Query routing in P2P systems has attracted much research interest in the last few years, with the aim of effectively and efficiently querying both multimedia [2] and semantic data [11]. As far as we know, no proposal exists which operates on both these kinds of data in an integrated approach.

As part of the NeP4B project, in this document we leverage our distinct experiences on semantic [6, 7] and multimedia [4, 5] query routing and propose to combine the approaches we presented in past works in order to design an innovative mechanism for a unified data retrieval in such a context. Two main aspects characterize our scenario. The former one is due to the heterogeneity of the peers' ontologies which may lead to semantic approximations during query reformulation. In this context, we pursue *effectiveness* by selecting, for each query, the peers which are semantically best suited for answering it, i.e. whose answers best fit the query conditions. The latter aspect is related to the execution of multimedia predicates, which is inherently costly (they typically require the application of complex functions to evaluate the similarity of multimedia features). In this setting, we also pursue *efficiency* by limiting the forwarding of a query towards the network's zones where potentially matching instances are more likely to be found, while pruning the others. In this way, we give the user a higher chance of receiving first the answers which better satisfy the query conditions.

The approach presented in this document relies on the matching and indexing technics presented in the Deliverable D4.2.1, which are necessary to rewrite a query and to address it towards the peer which most probably keep most relevant information. Moreover, through the document we will consider the Reference Scenario presented D4.2.1 and reported here in Figure 1.

To this end, we introduce a query processing model which satisfies the interoperability demands highlighted in the NeP4B project. The proposed model does not compel to a fixed semantics but rather it is founded on a *fuzzy* settlement which proved to be sound for a formal characterization of the approximations originated in the NeP4B network for both semantic [6] and multimedia data [12].

In this document we present:

- in Section 3 our semantic query routing mechanism, able to select the subnetworks which best satisfy the query conditions, which is guided by the semantic mappings between the peers described in D4.2.1;
- in Section 4 our routing and querying techniques for multimedia content, based on the indexing methodology presented in D4.2.1;
- in Section 5 our proposal to combine the above approaches in order to design an innovative mechanism which exploits the two main aspects characterizing the querying process in such a context: the semantics of the concepts in the peers' ontologies and the multimedia contents in the peers' repositories;
- in Section 6 different routing policies and how routing strategies influence the order of the returned answers. This allows different query processing approaches to be implemented, based on the specific consortium needs and policies;
- in Section 7 how semantic mappings are used to rewrite the query in order to obtain a rewriting using concepts from an ontology;

- in Section 8 different experiments on routing mechanism and in particular the first one is about the lonely SRI technic, the second one is performed on combined routing and the third one on different routing policies.

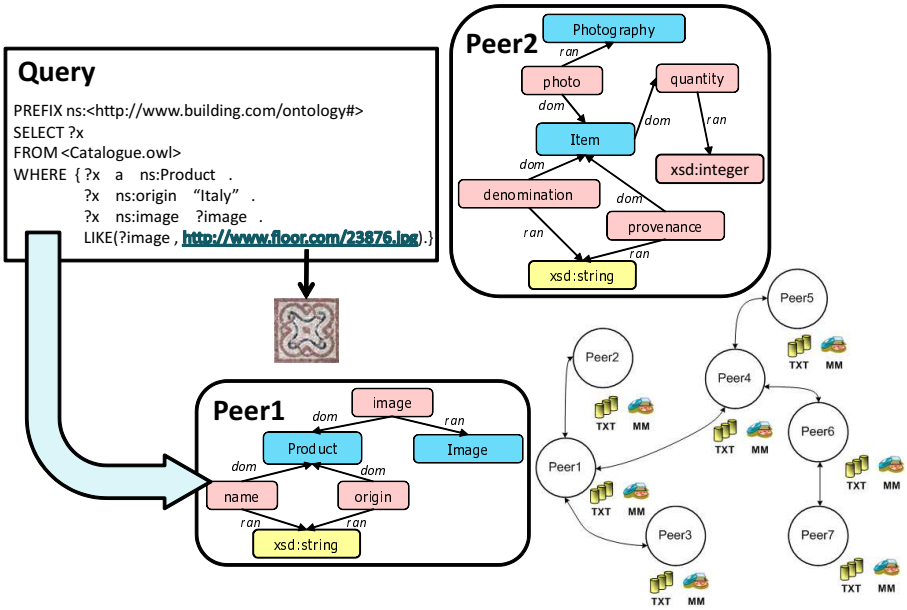



Figure 1: Reference scenario

# 3 Semantic Query Routing

An important aspect to be considered working in a PDMS is that it underlies a potentially very large network able to handle huge amounts of data. In this context, any relevant peer may add new answers to a given query and different paths to the same peer may yield different answers [14]. For this reason, *query routing* [8], i.e. the process of selecting the most promising peers, is a fundamental issue for querying distributed resources. In particular, in a semantic web perspective, a query posed over a given peer should be forwarded to the most relevant peers that offer semantically related results among its immediate neighbors first, then among their immediate neighbors, and so on.

When a query is forwarded through a semantic path, it undergoes a multi-step reformulation which may involve a chain of semantic approximations. To this purpose, our proposal is to exploit such approximations for selecting the direction which is *more likely* to provide the best results to a given query. Our perspective is knowledge-based, in that the routing of a query is guided by the semantic mappings between the peers.

Our proposal is a fully distributed framework which summarizes the cumulative loss of information of entire subnetworks, so that each peer can choose to forward a query towards the subnetwork(s) that minimizes the information loss. To this end, we follow an intensional approach in which we consider scores associated to semantic mappings to represent the semantic compatibility occurring between the involved portions of the peers' schemas. Scores are initially called *membership grades* or simply *similarity values* between concepts. They are used to quantify the relevance of peers' data to a query, and they are combined at each reformulation step to obtain an overall score representing the cumulative information loss of entire semantic paths.

Some kind of information about the relevance of the whole semantic paths should be available in the network, maintained up-to-date, and easily accessible for query routing purposes. To this end, we borrow the idea of a distributed-index mechanism from the literature which maintains indices at each node. In our proposal, a *Semantic Routing Index* (SRI) [8] summarizes, for each concept of its peer's schema, the semantic approximation "skills" of each subnetwork reachable from its immediate neighbors, and thus gives a hint of the relevance of the data which can be reached in each path. For instance, the Peer1's SRI will contain two entries, one for the upward subnetwork and one for the downward one. The semantic knowledge stored in a SRI is summarized on the available directions in order to maintain the size of the semantic index proportional to the number of neighbors, thus scaling well in a PDMS scenario.

## 3.1 Semantic Routing Indices

The membership grades given by generalized semantic mappings can be exploited for a wise propagation of a given query formulated over a peer $p$ towards the most promising direction in the network, i.e., towards $p$'s neighbors whose subnetworks are the most semantically related to the query. To this purpose, each peer $p$ maintains a matrix which contains the membership grades between its schema and the schemas of its neighbors. This matrix is used as a routing index and it is named *Semantic Routing Index (SRI)*. More precisely, if $p$ has $n$ neighbors, its SRI has $n + 1$ rows, where the first row refers to the knowledge on the local schema of peer $p$.

**Definition 1 (SEMANTIC ROUTING INDEX)** *Given a peer $p$ with schema $S = \{C_1, ..., C_m\}$ and neighbors $p_i ... p_n$ we can define the $p$'s SRI as a matrix of $n+1$ rows and $m$ columns, such that each entry $SRI[i][j]$, for $i = 1...n$ and $j = 1...m$, is the membership grade $\mu(C_j, C_j^{\triangle})$ of the instance $(C_j, C_j^{\triangle})$ of the generalized semantic mapping $M(S, S_i^{\triangle})$.*

A sample fragment of Peer1' SRI is represented in Figure 2, where, for instance, the score 0.34 in the Peer4 row and the *Product* column is the outcome of the aggregation of the scores associated to the paths Peer4, Peer4-Peer5, Peer4-Peer6 and Peer4-Peer6-Peer7.

The space required for storing a SRI at a peer is proportional to the number of the peer's neighbors, and thus quite modest. This makes our distributed-index mechanism scalable in a P2P context.

| $SRI_{Peer1}$ | Product | name | origin | ... |
|---|---|---|---|---|
| Peer1 | 1.0 | 1.0 | 1.0 | ... |
| Peer2 | 0.73 | 0.88 | 0.75 | ... |
| Peer3 | 0.69 | 0.48 | 0.30 | ... |
| Peer4 | 0.34 | 0.21 | 0.22 | ... |

Figure 2: Peer1's SRI.

## 3.2 Spreading Semantic Information in the Network

Since SRIs summarize the semantic information offered by the network, they change whenever the network itself changes. This may occur in response to either the joining/leaving of peers, or to changes in peers' schemas which possibly involve changes in the semantic mappings. We first focus our attention on the evolution of the PDMS's topology.

SRIs evolution is managed in an incremental fashion as follows. As a base case, the SRI of an isolated peer $p$ having schema $S$ is made of the single row $[1, \ldots, 1]$, i.e., it contains the membership grades of the concepts in $S$ in the self mapping $M(S, S)$. This row expresses the semantic approximation offered by the subnetwork rooted in $p$, yet made of the only peer $p$.

A simplification of the process of Peer1's SRI update when Peer1 connects to Peer4 is shown in Fig. 3 (P1 and P4 in the figure). When a peer connects to another peer, each one *aggregates* its own SRI $SRI$ by rows, according to an aggregation function $g$ (section 3.4.1). The result of this aggregation operation is a tuple $SRI^g = [\mu_1, \ldots, \mu_m]$. Each $\mu_j$ is the membership grade of concept $C_j$ in the schema $S$ of the peer to the fuzzy relation obtained by the aggregation of the SRI's rows, i.e., $\mu_j = g(SRI[0][j], \ldots, SRI[n][j])$ for $j = 1 \ldots m$. The so obtained fuzzy relation $SRI^g$ and the schema $S$ are then sent to the other peer.

After a peer, say $p_i$, receives such knowledge from the other peer, say $p_j$, a semantic mapping $M(S_i, S_j)$ is established between $S_i$ and $S_j$. Then, $p_i$ extends its SRI $SRI_i$ with a new row for $p_j$. The membership grades of this newly created row are obtained in two steps: 1) $M(S_i, S_j)$ is composed with the aggregated SRI provided by $p_j$ (section 3.4.2) to obtain a fuzzy relation which expresses the extension of the semantic paths originating from $p_j$ (represented by the aggregated SRI) with the connection between $p_i$ and $p_j$; 2) the so obtained fuzzy relation is then aggregated with $M(S_i, S_j)$ to include the semantic path connecting $p_i$

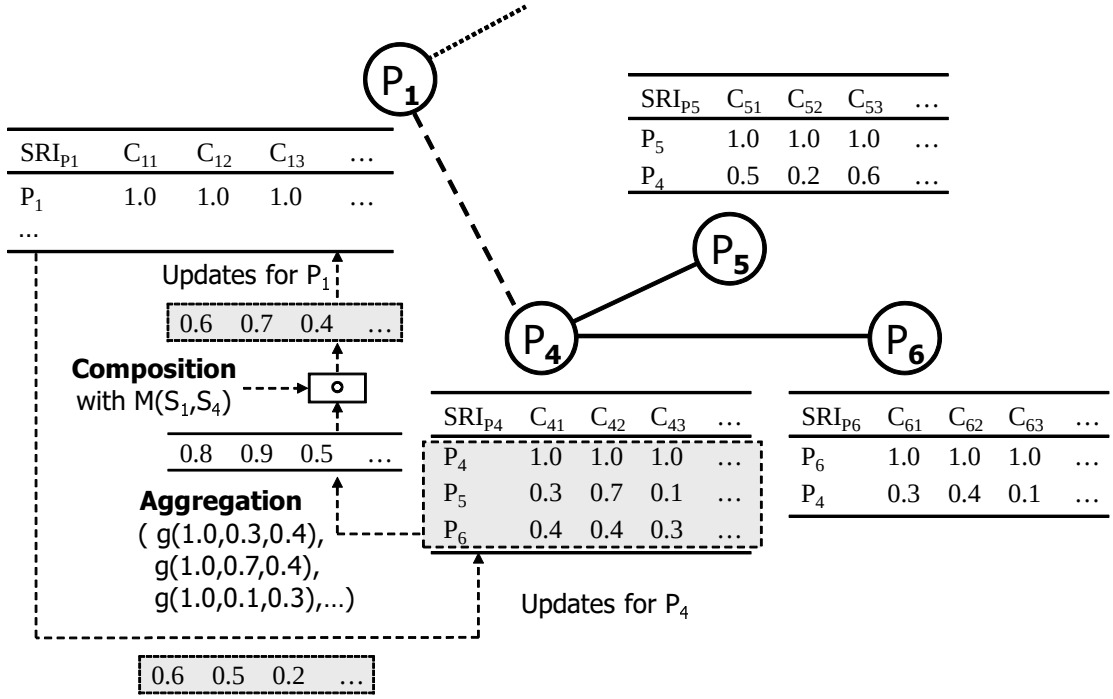



Figure 3: SRI evolution

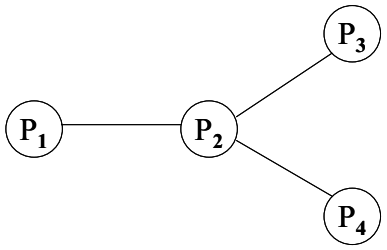



Figure 4: Simple network of connected peers

with $p_j$. More precisely,[1] $SRI_i[j][k] = g(M(S_i, S_j)(C_k, C'_k), M(S_i, S_j)(C_k, C'_k) \circ^I SRI^g[k])$, for $j = 1 \ldots m$.

Afterwards, both peers $p_i$ and $p_j$ need to inform their own reverse neighbors that a change occurred in the network and thus they have to update their SRIs accordingly. To this end, each peer, say $p_i$, sends to each reverse neighbor $p_{i_k}$ an aggregate of its SRI, excluding the $p_{i_k}$'s row, i.e., $\mu_j = g(SRI_i[0][j], \ldots, SRI_i[k-1][j], SRI_i[k+1][j], \ldots, SRI_i[n][j])$. When $p_{i_k}$ receives such aggregated information, it updates the $i$-th row of its SRI by recomputing the membership values as discussed above.

We now present a simple example of SRI construction process.

**Example 3.1** *Considering the simple network shown in Figure 4 and assuming that each peer $p_i$ has an ontology $Ont_i = c_{i,1}, ..., c_{i,m}$. We also define that a mapping between*

[1] Note that, because of the boundary condition of the t-norm $I$ used for composition, and being $g$ an idempotent function, the base case of connection of an isolated peer $p_j$ to a peer $p_i$ results in $SRI_i[j][k] = M(S_i, S_j)(C_k, C'_k)$.

*peer $p_i$ and a neighboring $p_j$ is $M(Ont_i, Ont_j)$ and a generalized semantic mapping is $M(Ont_i, Ont_j^{\triangle})$. Initially peers are isolated, i.e. they are disconnected each other so each of them builds an initial SRI having a single row, i.e. the self mapping $M(Ont_i, Ont_i)$ on its own concepts (no semantic approximation is present at this level). Then node $p_2$ connects to node $p_3$ and this implies that node $p_3$ sends its SRI, i.e. the self mapping $M(Ont_3, Ont_3)$, to node $p_2$. In this base case, $M(Ont_3, Ont_3)$ is nothing but $M(Ont_3, Ont_3^{\triangle})$, $Ont_3^{\triangle}$ being simply $Ont_3$. $p_2$ adds a new row to its SRI about the semantic approximation given by $p_3$. The mapping understood by this row, $M(Ont_2, Ont_3^{\triangle})$ is obtained by the composition of $M(Ont_2, Ont_3)$ and $M(Ont_3, Ont_3^{\triangle})$. Being $M(Ont_3, Ont_3^{\triangle})$ the identity mapping on $p_3$, the composition operation trivially reduces to $M(Ont_3, Ont_3)$. The same applies when $p_2$ connects $p_4$. In this case the new row added to $SRI_2$ understands $M(Ont_2, Ont_4)$. In summary, at the end of this process $SRI_2$ contains three rows, one for the self mapping and one for each mapping towards the neighboring peers $p_3$ and $p_4$. Then, let us suppose that node $p_1$ connects to node $p_2$. $p_2$ needs to aggregate its SRI before sending it to $p_1$. The mapping understood by the aggregated SRI is the generalized mapping $M(Ont_2, Ont_2^{\triangle})$ given by the union of the mappings understood by the three rows: $M(Ont_2, Ont_2)$, $M(Ont_2, Ont_3^{)}$ and $M(Ont_2, Ont_4)$. Finally $p_1$ combines $M(Ont_2, Ont_2^{\triangle})$ with its local mapping $M(Ont_1, Ont_2)$ to obtain the mapping $M(Ont_1, Ont_2^{\triangle})$ understood by the row concerning $p_2$'s subnetwork to be added to its SRI.*

As the values stored in the SRIs are computed incrementally, we have to show that they actually correspond to the membership values of the generalized semantic mappings.

**Theorem 1** *Whenever the aggregation function $g$ is associative and the composition function $I$ is distributive w.r.t the aggregation function $g$, the process described above is correct, i.e. when applied to the Semantic Routing Index of peer $p_i$, $SRI_i[j]$ is the generalized semantic mapping $M(S_i, S_j^{\triangle})$ between $p_i$ and $p_j$.*

The proof is by induction on the construction process and it relies on the properties of the composition and aggregation functions. Notice that in order to make the function $U$ associative, it suffices to maintain the number of aggregated paths as additional 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 SRI and then inform the remaining neighbors that a change on its own subnetwork has occurred by sending new aggregates of its SRI to them. A similar procedure applies in case of modifications of the semantic knowledge maintained at each peer, for instance when a new concept is added to the peer's schema. When many changes occur in the PDMS, a careful policy of updates propagation may be adopted. For instance, when changes has a little impact on its SRI, a peer may also decide not to notify the network. This would reduce the amount of exchanged messages as well as the computational costs due to SRI manipulation. We are aware that the definition of such policies, recommended for highly dynamic PDMSs, is a fundamental issue, so we plan to deal with it in future work.

## 3.3 SRIs for Query Routing

When a peer $p$ needs to forward a query $q$, it accesses its own SRI for determining the neighboring peers which are most semantically related to the concepts in $q$. For the sake

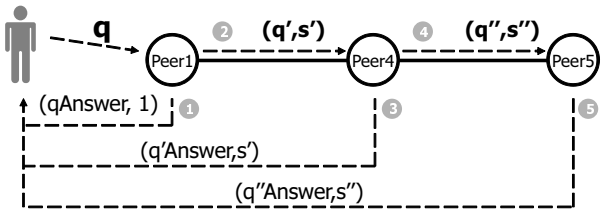



Figure 5: Query answering along a semantic path

of clarity, we start simple, and we assume the query $q$ refers to a single concept C. The choice of the semantically best neighboring peers is done by evaluating the column of its SRI corresponding to $C$. In particular, the highest values in this column make the corresponding neighbor to be the selected peers. For instance, considering the concept *Product* of the query in Figure 1, the most promising subnetwork would be the one rooted at Peer2 (score 0.73 in Figure 2).

In the general case of a complex query involving more concepts, the choice of the best neighbors is given by applying scoring rules which, for each neighboring peer $p_i$, combines the corresponding grades in the SRI (section 3.4.3) for all the corresponding concepts in $q$.

Once the best neighboring peer $p_i$ is found, the semantic mapping $M(S, S_i)$ is exploited to unfold the query $q$ in $q'$. $q'$ is then routed towards the subnetwork $p_i^{\triangle}$, where, starting from $p_i$ which in turn evaluates $q'$ and returns its local results to $p$, the process possibly reiterates. Figure 5 schematizes the query execution process on a possible path, *Peer1-Peer4-Peer5*, of the reference example.

## 3.4 Scoring foundations

As already said, our propose algorithm is founded on a fuzzy settlement which provides a formal semantics of the approximations originated by the heterogeneity of the schemas in a PDMS. Fuzzy set theory has been widely applied in contexts where uncertainty of description is intrinsic in the nature of the data [12]. In this context, to model the uncertainty of the information we deal with, scores can be properly viewed as *fuzzy grades* while the peer $p$'s schema concepts in $M_p$ and $SRI_p$ as fuzzy sets.

### 3.4.1 The aggregation function

The aggregation function $g$ should be chosen conveniently to model the semantic aggregation of semantic grades. First, the following properties, which express the essence of the notion of aggregation [13], must hold:

1. *g* is *monotonic increasing* in all its arguments;
2. *g* is a *continuous* function;
3. *g* respects the *boundary conditions* $g(0, ..., 0) = 0$ and $g(1, ..., 1) = 1$;
4. *g* is a *symmetric* function for all its arguments;
5. *g* is a *idempotent* function, that is, $g(a, ..., a) = a \forall a \in [0, 1]$.

Then, several choice are possible for *aggragate* satisfying the above properties, for instance functions such as the min, the max, any generalized mean or any ordered weighted averaging function.

### 3.4.2 The composition function

A composition function should be representative of an *accumulation of semantic approximations*. In its essence, composition operations on fuzzy sets are usually modeled by means of a Triangular-norm (T-norm) $i$, which is a binary function on the unit interval that satisfies the boundary condition (i.e. $i(s,1) = s$), as well as the monotonicity, the commutativity and the associativity properties. In particular, the latter is necessary to compose fuzzy sets along semantic paths. Examples of frequently used T-norms are:

- *standard intersection*:
$$i(a,b) = min(a,b)$$
- *algebraic product*:
$$i(a,b) = ab$$
- *bounded difference*:
$$i(a,b) = max(0, a+b-1)$$
- *drastic intersection*:
$$i(a,b) = \begin{cases} a & \text{when } b = 1 \\ b & \text{when } a = 1 \\ 0 & \text{otherwise} \end{cases}$$

In order to obtain the effect of semantic attenuation of grades between long distance peers, the monotonicity property is not enough and thus not all T-norms are good for our purposes. For instance, a good T-norm is the algebraic product whereas the min function is not.

### 3.4.3 The combination function

A query formula $f$ can be expressed as $f ::= p \mid f \wedge f \mid f \vee f \mid (f)$, where $p$ can be either a selected schema concept or a relational predicate comparing a concept value to a constant or a relationship-type predicate expressing a semantic relationship condition between a pair of schema concepts.

Given a set of concepts involved in a query formula $f$ and one similarity score for each concept, the overall score of $f$ through the *combination* function is obtained by combining the concepts' scores according to the logical connectives if $f$. More in details, when dealing with the conjunction of predicates, the fuzzy grades of predicates are to be combined according to a fuzzy intersection semantics $f_\wedge(s,s')$ which is usually modeled by means of a fuzzy T-norm. In a similar function, when dealing with disjunction of predicates, the union of two fuzzy sets $f_\vee(s,s')$ is usually modeled by means of a fuzzy Triangular-conorm $u$ (also called T-conorm), which is a binary function on the unit interval that satisfies the boundary condition (i.e. $u(s,0) = s$), as well as the monotonicity, the commutativity and the associativity properties. Examples of frequently used T-conorms are:

- *standard union*:
$$u(a,b) = max(a,b)$$
- *algebraic sum*:
$$u(a,b) = a + b - ab$$
- *bounded sum*:
$$u(a,b) = min(1, a+b)$$
- *drastic union*:
$$u(a,b) = \begin{cases} a & \text{when } b = 0 \\ b & \text{when } a = 0 \\ 1 & \text{otherwise} \end{cases}$$

# 4 Multimedia Query Routing

As presented in the Deliverable D4.2.1 about the mapping and the indexing, multimedia content are treated by means of a special routing index (RI), called Multimedia Routing Index (MRI) specifically thought for unstructured P2P networks. No organization in special shapes are required, in contrast to the DHT-based networks. Nonetheless, particular P2P topologies may pose problems for RIs. Each MRI is associated with a network link, and collect and summarize information about objects located in the portion of the network that can be reached along this link. If the network present a loop, the information about the peers involved in the loop could be duplicated. To avoid this problem, we use a node caching mechanism to avoid repeated updates of our indices. The final effect is that we can consider the network topology of our system as a *logical tree*. Accordingly, the MRI associated with a link $p \rightarrow p_j$ represents the values of all the objects stored in the logical *sub-tree*, rooted on $p_j$ and denoted by $\mathcal{T}(p \rightarrow p_j)$.

## 4.1 Query Processing

The query process starts when a peer receives a query from a user. We assume that queries could be originated from any node $p$ in the system and could involve an arbitrary number of features. The query processes necessitate of the index described in Section 5 of Deliverable D4.2.1. As explained, the data indexing process is intended to produce a summarized description of all the features of the objects owned by each peer in order to produce efficient multi-feature MRIs. The aim of these indices is to provide a concise but yet sufficiently detailed description of the resources available in a given network area. This information is then used at query-resolution time to prune entire system zones from searching, thus easing the search process.

Let $Q$ be a query and $r$ the query radius. Since $Q$ is defined over a set of attributes, its radius is obtained as a linear combination of the radiuses of the sub-queries over each single feature. Thus, we have that $r = \alpha_1 r_1 + \ldots + \alpha_L r_L$, where $L$ is the number of features requested by $Q$.

In order to compare $Q$ with the MRIs defined in the previous section, $Q$ is translated into a set of bit vectors. For each single feature $F_h$ and its associated set of reference objects $R^{F_h} = \{R_1^{F_h}, \ldots, R_m^{F_h}\}$, we consider the interval defined by $[d(Q^{F_h}, R_i^{F_h}) - r_j, d(Q^{F_h}, R_i^{F_h}) + r_j]$, where $r_j$ is the radius of $Q^{F_h}$ with respect to $F_h$. A $k-$bit vector is constructed by setting to 1 all the entries that correspond to intervals that are covered (even partially) by the requested interval. All the other entries are set to 0. Such an index is denoted as ***QueryIdx*** $\left(Q^{F_h}\right)_{R_i^{F_h}}$.

These indices can be directly compared with the MRI ones, in order to find which are the neighbors connected with the network zones with the highest chance to contain relevant objects. The matching phase is performed in the following way. Let $p$ be the peer that receives a query $Q$, ***QueryIdx*** $\left(Q^{F_h}\right)_{R_i^{F_h}}$ the set of indices of the query relative to the feature $F_h$ and ***LinkBitIdx*** $(p \rightarrow p_j, F_h)$ the MRIs on the same feature for a peer $p_j \in$ ***Nb*** $(p)$. For each reference point $R_i^{F_h}$ of $F_h$, the indices ***QueryIdx*** $\left(Q^{F_h}\right)_{R_i^{F_h}}$ and ***LinkBitIdx*** $(p \rightarrow p_j, F_h)_{R_i^{F_h}}$ are compared. A match is considered to happen if and only if there exists at least one entry $e$ for which both indices have a value greater than zero. In order to forward $Q^{F_h}$ from $p$ to $p_j$, all the indices of all the reference points of all the features must match the query corresponding ones.

Since the MRIs can also be regarded as histograms of the distribution of objects with reference to every feature, it is also possible to know the number of potentially matching objects. This possibility gives us the ability to perform different query forwarding strategies. One of these strategies consists in allowing a peer $p$, which has to forward a query, to select its neighbors on the basis of the number of potential matchings they have in their subnetworks. The selection is made using what we call the *requested coverage*, i.e. the neighbors the query is to be forwarded to must have a number of matchings that is equal or above a given percentage of all the possible matchings given by the MRIs. The aim of this process is to further reduce the number of query messages, while trying to collect the largest number of results.

## 4.2 Query Processing Example

Suppose that we have a network like the one depicted in Figure 7. In this network, each peer has a local repository formed by objects like the one in Figure 6. The indices shown in Figure 7 are the routing indices has they are seen by the parents of each node.


Figure 6: Example mixed multimedia object in NeP4B.

Each peer has its own local index for each reference object like the one in Figure 8. Suppose that a user poses the query shown in Figure 9 and that the *requested coverage* is set to 50%. forwarding phase. For the sake of simplicity, only matching indices with their relative number of potential matchings are shown. As can be seen, $p_1$ and $p_2$ have matching indices. Thus, $p_3$ is excluded from the forwarding process. Moreover, $p_1$ is the neighbor with the highest number of matchings. It covers $3/4$ of all potential matching objects. Since this percentage is above the requested coverage, the query is forwarded only to $p_1$. receives the query from, since it is the neighbor with the most relevant set of potential matchings. $p_1$, in turn, search in its local repository and send the query to $p_4$. Finally, both $p_1$ and $p_4$ give their answers back to $p$.

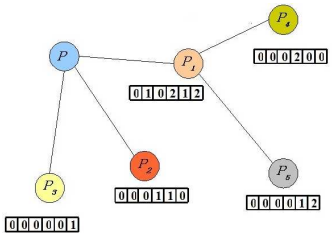



Figure 7: Example of a network

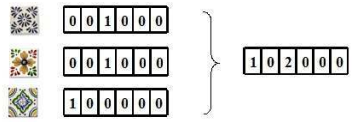


Figure 8: Peer local index

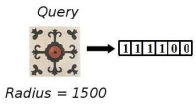



Figure 9: Index of a range query

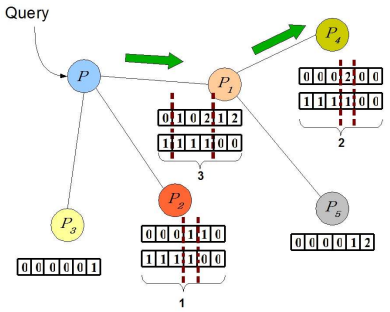



Figure 10: Index of a range query

## 4.3 Simulation Results

In this section we present the results obtained by a simulation of our system. The aim of this simulation experiments is to the number of nodes that receives a query because their indexes show that they have potential matchings for the query. Simulation results were computed as intervals with $90\%$ confidence level and each measurement was repeated multiple times in order to get confidence intervals having width of less than $5\%$ of the central value. The simulation model has been implemented using the C++ library described in [10]. The simulation settings are as follows: The network topology is a tree generated randomly, with at most five children per node; We use a network with 204 nodes that represent an equal number of clusters of the available data; The object used in the network are 159805 real images taken from the Flickr archive (http://www.flickr.com). The clusters are not disjunct, i.e., although each data object is assigned univocally to only one peer, the space supervised by each peer may overlap with the one owned by another peer. Each object is characterized by two MPEG-7 standard features, scalable color and edge histogram. We used 10 different reference objects for constructing the indices.They are chosen as 10 random objects taken from the 10 peers with the largest local repositories.

Fig. 11 illustrates the nodes involved in the forwarding of queries that use both the feature at the same time. The combined search involves from about $12\%$ to $20\%$ percent of the total resources. The fact that the number of nodes involved exceeds the number of the resource that exactly match the queries is due to false positives derived from the approximation involved in the index creation. We have compared the performances of the combined search with the ones of the searches for each single feature, using the same network and the same radii used for the combined queries (see 12(a) and 12(b)). The results show that the use of the indices allow to pruning from searching uninteresting network areas both for multi- and single-feature queries. Thus, the MRIs achieve the primary goal of creating an efficient routing system that could exploit all the features of the objects, singularly or in a combined manner. As can be seen, the result of the combined search is better of that of each single resource. This is due to the fact that the use of the same radii in the combined searches led

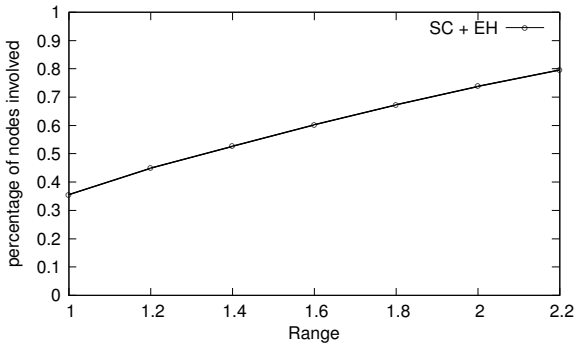
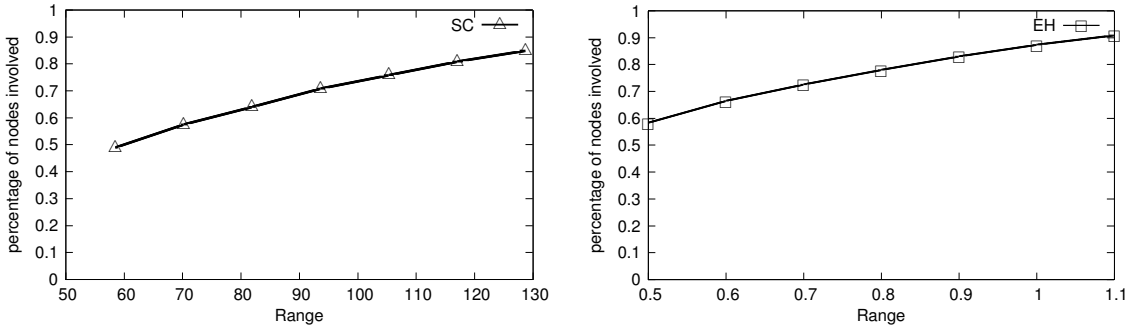



Figure 11: Nodes involved in the combined range query

to more selective queries. Thus the data and the peers involved are fewer than that each single feature query.




(a) SC Range Queries

(b) EH Range Queries

Figure 12: Nodes involved in the propagation of single-feature range queries

# 5 Combined Query Routing

As part of the Project, we leverage our distinct experiences on semantic [8, 7] and multimedia [4] query routing and propose to combine the approaches we presented in past works in order to design an innovative mechanism which exploits the two main aspects characterizing the querying process in such a context: The semantics of the concepts in the peers' ontologies and the multimedia contents in the peers' repositories. More precisely, since the reformulation process may lead to some semantic approximation, we pursue effectiveness by selecting, for each query, the peers which are semantically best suited for answering it. Further, since the execution of multimedia similarity queries is inherently costly (they typically require the application of complex distance functions) we also pursue efficiency by limiting their forwarding to the network's zones where potentially matching objects could be found, while pruning the others.

This combination procedure is a delicate issue because SRI and MRI are orthogonal approaches as for, as already said, objectives but also for routing type and query processing. Indeed, while SRI routing bases on schema structure and so it is intensionally driven, the MRI routing is extensionally driven and as for SRI a query rewriting is necessary while MRI doesn't need for it.

## 5.1 Combined Query Answering

Whenever a peer $p$ receives a query, both the semantic and the multimedia routing approaches associate each $p$'s neighbor a score quantifying the semantic relevance and the percentage of potential matching objects in its subnetwork, respectively. This allows $p$ to rank its own neighbors w.r.t. their ability to answer a given query effectively, i.e. minimizing the information loss due to its reformulation along semantic mappings, and efficiently, i.e. minimizing the network load due to the exploration of useless subnetworks.

Thus, since both the semantic and multimedia scores induce a total order, they can be combined by means of a proper aggregation function in order to obtain a global ranking. In the following we first define how these score are obtained from the query posed on the ontology of the queried peer. Query conditions are expressed using predicates that can be combined in logical formulas through logical connectives, according to the syntax:

$$
\begin{aligned}
f &::= \langle triple\_pattern \rangle \langle filter\_pattern \rangle \\
\langle triple\_pattern \rangle &::= triple \mid \langle triple\_pattern \rangle \wedge \langle triple\_pattern \rangle \\
\langle filter\_pattern \rangle &::= \varphi \mid \langle filter\_pattern \rangle \wedge \langle filter\_pattern \rangle \mid \\
&\quad \langle filter\_pattern \rangle \vee \langle filter\_pattern \rangle \mid (\langle filter\_pattern \rangle)
\end{aligned}
$$

where $triple$ is an RDF triple and a filter $\varphi$ is a predicate where relational ($=$, $<$, $>$, $<=$, $>=$, $\neq$) and similarity ($\sim_t$) operators operate on RDF terms and values. In particular, note that $\sim_t$ refers to multimedia content and translates the `LIKE` operator where $t$ is the specified similarity threshold.

Each peer receiving a query first retrieves the answers from its own local data then it reformulates the query towards its own neighborhood.

The evaluation of a given query formula $f$ on a local data instance $i$ is given by a score $s(f, i)$ in $[0, 1]$ which says how much $i$ satisfies $f$. The value of $s(f, i)$ depends on the evaluation on $i$ of the filters $\varphi_1, \ldots, \varphi_n$ that compose the filter-pattern of $f$, according to a scoring

function $sfun_\varphi$, that is: $s(f(\varphi_1, \ldots, \varphi_n), i) = sfun_\varphi(s(\varphi_1, i), \ldots, s(\varphi_n, i))$. Note that filters are predicates of two types: relational and similarity predicates. A relational predicate is a predicate which evaluates to either 1 (true) or to 0 (false). The evaluation of a similarity predicate $\varphi$ follows instead a non-Boolean semantics and returns a score $s(\varphi, i)$ in $[0, 1]$ which denotes the grade of approximation of the data instance $i$ with respect to $\varphi$. It is set to 0 when the similarity of $i$ w.r.t. the predicate value is smaller than the specified threshold $t$, and to the grade of approximation measured, otherwise. The scoring function $sfun_\varphi$ combines the use of a t-norm ($s_\wedge$) for scoring conjunctions of filter evaluations, and the use of a t-conorm ($s_\vee$) for scoring disjunctions. A t-norm (resp., t-conorm) is a binary function on the unit interval that satisfies the boundary condition (i.e. $s_\wedge(s, 1) = s$, and resp., $s_\vee(s, 0) = s$), as well as the monotonicity, the commutativity, and the associativity properties.[2] The use of t-norms and t-conorms generalizes the query evaluation model with respect to the use of specific functions. Therefore, for a given peer $p$, the query answers retrieved from the evaluation of $f$ on its own local data is given by $Ans(f, p) = \{(i, s(f, i)) \mid s(f, i) > 0\}$, i.e., it is the set of local data instances which satisfy $f$, possibly with a certain grade of approximation.

Due to the heterogeneity of schemas, any reformulation a peer $p_i$ performs on a given query formula $f$ towards one of its neighbors, say $p_j$, gives rise to a semantic approximation which depends on the strength of the relationship between each concept in $f$ and the corresponding concept in $O_j$. Such an approximation is quantified by a scoring function $sfun_c$ which combines the $p_j$'s mapping scores on $f$'s concepts: $s(f, p_j) = sfun_c(\mu(C_1, C'_1), \ldots, \mu(C_n, C'_n))$ where $C_1, \ldots, C_n$ are the concepts in $O_i$ involved in the query formula, and $C'_1, \ldots, C'_n$ are the corresponding concepts in $O_j$ according to $M(O_i, O_j)$. $sfun_c$ is a t-norm as all the involved concepts are specified in the $triple\_pattern$ of $f$ and triples can only be combined through conjunctions.

Starting from the queried peer, the system can access data on any peer in the network which is connected through a semantic path of mappings. When the query is forwarded through a semantic path, it undergoes a multi-step reformulation which involves a chain of semantic approximations. The semantic approximation given by a semantic path $p_1 \rightarrow p_2 \rightarrow \ldots \rightarrow p_m$ (in the following denoted as $P_{p_1 \ldots p_m}$), where the submitted query formula $f_1$ undergoes a chain of reformulations $f_1 \rightarrow f_2 \rightarrow \ldots \rightarrow f_m$, can be obtained by composing the semantic approximation scores associated with all the reformulation steps: $s(f_1, P_{p_1 \ldots p_m}) = sfun_r(s(f_1, p_2), s(f_2, p_3), \ldots, s(f_{m-1}, p_m))$, where $sfun_r$ is a t-norm which composes the scores of query reformulations along a semantic path of mappings.

Summing up, given a query formula $f$ submitted to a peer $p$, the set of accessed peers $\mathcal{P}' = \{p_1, \ldots, p_m\}$,[3] and the path $P_{p \ldots p_i}$ used to reformulate $f$ over each peer $p_i$ in $\mathcal{P}'$, the semantics of answering $f$ over $\{p\} \cup \mathcal{P}'$ is the union of the query answers collected in each accessed peer: $Ans(f, p) \cup Ans(f, P_{p \ldots p_1}) \cup \ldots \cup Ans(f, P_{p \ldots p_m})$ where each answer $Ans(f, P_{p \ldots p_i})$ contains the set of the results collected in the accessed peer $p_i$ together with the semantic approximation given by the path $P_{p \ldots p_i}$: $Ans(f, P_{p \ldots p_i}) = (Ans(f, p_i), s(f, P_{p \ldots p_i}))$. As observed before, as far as the starting queried peer is involved, no semantic approximation occurs as no reformulation is required (i.e. $s(f, p) = 1$).

[2] Examples of t-norms are the $min$ and the *algebraic product* operators, whereas examples of t-conorms are the $max$ and the *algebraic sum* operators.

[3] Note that $\mathcal{P}'$ not necessarily covers the whole network (i.e., $\mathcal{P}' \subseteq \mathcal{P}$).

## 5.2 Semantic Scores

When a peer $p$ receives a query formula $f$, it exploits its SRI scores to determine a ranking $R^p_{sem}(f)$ for its neighborhood, so as to identify the directions which best approximate $f$. More precisely, for each neighbor $p_i$ an overall score is computed by combining, by means of the scoring function $sfun_c$, the scores the SRI row $SRI[p_i]$ associates with the concepts $C_1, \ldots, C_n$ in $f$: $R^p_{sem}(f)[p_i] = sfun_c(\mu(C_1, C_1^{\Delta}), \ldots, \mu(C_n, C_n^{\Delta}))$. Intuitively, the higher is the overall score, the more likely the peer's subnetwork will provide semantically relevant results to the query.

## 5.3 Multimedia Scores

When a peer $p$ receives a multimedia query expressed by the formula $f$ containing a `LIKE` predicate, instead of flooding the network by forwarding $f$ to all its neighbors, $p$ matches the indices of $Q$ with the corresponding routing indices of its neighborhood. The matching phase outputs a score that suggests the degree of relevance of the query with respect to each of the possible forwarding directions. More precisely, $p$ determines a ranking $R^p_{mm}(f)$ for its neighborhood, by taking the minimum of the products (element by element) of the indices $\mathit{QueryIdx}(Q)_{R_k}$ and $\mathit{LinkBitIdx}(p, p_i^{\Delta})_{R_k}$ for each neighbor $p_i \in \mathit{Nb}(p) = \{p_1, \ldots, p_n\}$ and each reference object $R_k$, and then evaluating the following ratio:

$$R^p_{mm}(f)[p_i] = \frac{\sum_{h=1..L} \alpha_h \min_k \left[\mathit{QueryIdx}\left(Q^{F_h}\right)_{R_j^{F_h}} \cdot \mathit{LinkBitIdx}(p \rightarrow p_j, F_h)\right]}{\sum_{h=1..L} \alpha_h \sum_{j=1..n} \min_k \left[\mathit{QueryIdx}(Q^{F_h})_{R_k^{F_h}} \cdot \mathit{LinkBitIdx}(p \rightarrow p_j, F_h)\right]} \quad (1)$$

$R^p_{mm}(f)[p_i]$ gives an intuition of the percentage of potential matching objects underneath the subnetwork rooted at $p_i$ with respect to the total objects retrievable through all the peers in $\mathit{Nb}(p)$.

## 5.4 Combined scores

More precisely, given the two distinct rankings $R^p_{sem}(f)$ and $R^p_{mm}(f)$ computed for the query formula $f$ on peer $p$ we need an *aggregation function* $\oplus$ which, when applied to $R^p_{sem}(f)$ and $R^p_{mm}(f)$, provide a $R^p_{comb}(f)$ reflecting the overall goodness of the available subnetworks: $R^p_{comb}(f) = \alpha \cdot R^p_{sem}(f) \oplus \beta \cdot R^p_{mm}(f)$, where $\alpha$ and $\beta$ can be set in order to give more relevance to either the semantic or multimedia aspect.

In [3] it is stated that optimal aggregation algorithms can work only with monotone aggregation function. Typical examples of these functions are the min and mean functions (or the sum, in the case we are not interested in having a combined grade in the interval $[0, 1]$). As an example of how the aggregation process works, let us go back to the sample query of our main scenario and suppose Peer1 obtains the scores in the following table.

| | $\text{SRI}_{\text{Peer1}}$ | $\text{MRI}_{\text{Peer1}}$ | $min()$ |
|---|---|---|---|
| Peer2 | 0.70 | 0.53 | 0.53 |
| Peer3 | 0.63 | 0.76 | 0.63 |

The rankings computed through SRI and MRI are Peer2-Peer3 and Peer3-Peer2, respectively. If we use the standard fuzzy conjunction $min$, we compute the following final ranking:

Peer3-Peer2. As a result, the most promising subnetwork will be the one rooted at neighbor Peer3.

The obtained ranking reflects the foreseen subnetworks ability in solving the received query both at schema (SRI-based information) and at multimedia (MRI-based information) levels and can thus be properly tailored in order to implement clever routing strategies. This is the subject of the following section.

# 6 Routing Policies

Starting from the queried peers, the objective of any query processing mechanism is to answer requests by navigating the network until a stopping condition is reached. A query is posed on the schema of the queried peer and is represented as a tuple $q = (id, f, sim, t)$ where: $id$ is a unique identifier for the query; $f$ is a formula of predicates specifying the query conditions and combined through logical connectives; $sim$ is the semantic approximation associated with the semantic path the query is following, i.e. the accumulated approximation given by the traversal of semantic mappings between peers' schemas; $t$ is an optional relevance threshold. When $t$ is set, those path whose relevance $sim$ is smaller than $t$ are not considered for query forwarding.

## 6.1 Query execution models

Query execution can be performed following different strategies, according to two main families of navigation policies [7]: The *Depth First* (*DF*) query execution model, which pursues efficiency as its objective, and the *Global* (*G*), or *Goal-based* model, which is designed for effectiveness. Both approaches are devised in a distributed manner through a protocol of message exchange, thus trying to minimize the information spanning over the network. For both models we experienced various routing strategies, which will be the subject of the next sections. In general, the models work in the following way: Starting from the queried node, a peer $p$, univocally identified as $p$.ID, receives query $q$ and a list $L$ of already visited nodes from the calling peer $C$ through an "execute" message. Then, it performs the following steps: 1. Accesses its local repository for query results; 2. decides a neighbor $Next$ among the unqueried ones which forward the query to; 3. reformulates the query $q$ into $q_{Next}$ for the chosen peer, using the semantic mappings $M_{Next}$ towards it; 4. forwards the query $q_{Next}$ to the neighbor $Next$ and waits for a finite response from it (see Figure 5). In order to accomplish the forwarding step, the semantic and multimedia information stored at the peer's SRI and MRI are used in a combination manner to rank the neighbors, thus also taking into account the relevance of results which are likely to be retrieved from their own subnetworks. More precisely, as we explained in Section 5, when a peer $p$ receives a query $q$ it exploits its indices information to compute a ranking $R^p_{comb}(f)$ on its neighbors expressing the goodness of their subnetworks w.r.t. the query formula $f$. In particular, in order to limit the query path, the DF model only performs a local choice among the $p$'s neighbors, whereas the G model adopts a global ranking policy among the already known and unvisited peers. Once the most promising neighbor has been chosen, the query $q$ is reformulated (through unfolding [14]) into $q_{Next}$. Then, $q_{Next}$ is assigned the semantic approximation value obtained by composing the semantic approximation obtained so far, i.e. $q.sim$, and the approximation for $q.f$ given by the semantic mapping $M_{Next}$ between the current peer and the neighbor $Next$, thus instantiating $q_{Next}.sim$. Finally, the query execution process can start backtracking, returning the control to the calling peer through a "restore" message. For this reason, $NS$, an array of navigation states, one for each query processed by $p$, is maintained at each node for query restoring. As we will see in detail, while the DF model performs backtracking only when a "blind alley" is reached, the G model constantly exploits it in order to reach back potentially distant goals.

```
executeDF(q,C,L):
00   localExecution(q,L);
01   NeighborList = ∅;
02   for each neighbor N of p except C
03     Srel = combine(q.f,SRI[N]);
04     Mrel = combine(q.f,MRI[N]);
05     relevance = aggregate(Srel, Mrel);
06     add (N,relevance) to NeighborList;
07   PQ = order NeighborList in desc
       order of relevance;

     restoreDF(id,L):
08   do
09     (Next,relevance) = pop(PQ);
10     prepareNextQuery(q,q_Next);
11   while (Next not NULL and
           (Next ∈ L OR q_Next.sim <= q.t));
12   if (Next is NULL)
13     delete NS[q.id];
14     send message restoreDF(q.id,L) to C;
15   else
16     NS[q.id] = (q,PQ,C);
17     send message executeDF(q_Next,p.ID,L)
             to Next;

localExecution(q,L):
00   add p.ID to L;
01   execute q on local repository;
02   if (stopping condition is reached)
03     stop execution

prepareNextQuery(q,q_Next):
00   q_Next.id = q.id;
01   q_Next.f = reformulate(q.f,M_Next);
02   q_Next.sim = compose(q.sim,combine
                  (q.f, M_Next));
03   q_Next.t = q.t;
```

```
executeG(q,C,L,GL):
00   localExecution(q,L);
01   for each neighbor N of p except C
02     Srel = fScore(compose(q.sim,
                     combine(q.f,SRI[N])));
03     Mrel = combine(q.f,MRI[N]);
04     relevance = aggragate(Srel,Mrel);
05     add (p.ID,N.ID,relevance,true) to GL;
06     order GL in desc order of relevance;

     restoreG(id,L,GL):
07   do
08     NextG = top(GL);
09   while (NextG not NULL and NextG.active
         = true and NextG.to ∈ L);
10   if (NextG is NULL)
11     stop execution
12   else if (NextG.active = false
           OR NextG.relevance <= q.t)
13     clean(p.ID,GL,NS);
14     send message restoreG(q.id,L,GL) to C;
15   else if (NextG.from <> p.ID)
16     clean(p.ID,GL,NS);
17     NH = selectNextHop(GL);
18     send message restoreG(q.id,L,GL) to NH;
19   else
20     prepareNextQuery(q,q_NextG.to);
21     NS[q.id] = (q,C);
22     NextG.active = false;
23     send message executeG(q_Next,p.ID,L,GL)
           to Next;

clean(id,GL,NS):
00   for each goal G in GL
01     if (G.from = id and G.active = true)
02       return;
03     else if (G.to = id)
04       target = G;
05   delete target from GL;
06   delete NS[id];
```

Figure 13: Query execution algorithms for “DF” (left) and “G” (right) routing models

## 6.2 The DF query execution model

The DF query execution model provides a routing index-oriented depth first visiting criteria. The left portion of Figure 13 shows the algorithm in detail. Once a peer receives an $\texttt{executeDF}(q, C, L)$ message, step 1 is performed through $\texttt{localExecution}(q, L)$), step 2 through lines 01-09, step 3 through $\texttt{prepareNextQuery}(q, q_{Next})$ and step 4 through lines 13-17. Notice that only $L$ “surfs” the network along with the query which, at each step, is locally instantiated on the peer’s dictionary.

In order to select a peer (lines 02-07), the combined routing $R^p_{comb}(f)$ must be evaluated. The evaluation of the semantic relevance of each neighbor $N$ (line 03) is performed by combining, according to a fuzzy logic approach, the similarity values in the SRI’s row associated to $N$, i.e. $SRI[N]$, and corresponding to each concept $c$ in the formula $q.f$: $\texttt{combine}(q.f, SRI[N])$. For instance, following our reference example in Figure 2 and dealing conjunction with the minimum, the combined score for a query on Peer1 towards Peer2 involving concepts “product” and “name” connected through AND would be $min(0.73, 0.88) = 0.73$. Similarly the multimedia relevance of each neighbor (line 04) is performed by combining values in the MRI’s row associated to $N$ ($MRI[N]$), and corresponding to each feature used to categorize data objects request in $q$. Where the row $MRI[N]$ refers to the index defined in D4.2.1 *LinkBitIdx* $(p \rightarrow p_j, F_h)_{R_i^{F_h}}$, with $j \equiv N$. We have said always $q.f$ but, while for semantic approach $q.f$ expresses concepts involved in the query, for multimedia one $q.f$ represents features used to categorized data objects with those MRIs. The two peer rankings, one relating to semantic relevance and one relating to multimedia rele-

vance of each neighbor, are combined, using an aggregation function (line 05) in order to perform a combined routing, as already seen in section 5. So the peer produces a unique list NeighborList and such list is then ordered on the basis of the peers' relevance (producing a priority queue $PQ$, line 07) such that the top neighbor roots the subnetwork with the best approximation skills. $L$ is used for cycle detection, i.e. in order to avoid querying the same peer more times (line 11). The forwarding process starts backtracking when the list of unvisited neighbors for the current peer is empty, returning the control to the calling peer (lines 12-14). When a peer receives the message `restoreDF(`$id$`,`$L$`)`, it reactivates the navigation state $NS = (q, PQ, C)$ for the query identifier $q.id$ and then follows lines 08-17 either to select the next peer which to forward the query to, or to continue backtracking. Thus, DF progresses by going deeper and deeper until the stopping condition is verified.

Based on the DF model, we devised the two following routing policies:

- **DF policy**: the "standard" depth-first policy, straightly implementing the DF model;
- **DFF (Depth-First Fan) policy**: a variation of DF, performing depth-first visit with an added twist. Specifically, at each node, DFF performs a "fan" by exploring all the neighbors having similarity above threshold $t$, then it proceeds in depth to the best subnetwork, as DF does. DFF is an attempt to enhance DF, as it tries to capture in less hops more answers coming from short semantic paths and, thus, being potentially more relevant than those retrieved by DF.

### 6.3 The G query execution model

Along with the DF model, we devised an additional one sharing some common principles but having effectiveness as its primary goal: The right part of Figure 13 shows the G model in detail. Differently from the DF one, in the G model each peer chooses the best peer to forward the query to in a "global" way: It does not limit its choice among the neighbors but it considers all the peers already "discovered" (i.e. for which a navigation path leading to them has been found) during network exploration and that have still not been visited. This is mainly achieved by managing and passing along the network an additional structure, called *Goal List* ($GL$), which is the evolution of DF's NeighborList (and $PQ$) and is a global ordered list of *goals*. Each goal $G$ contains information useful for next peer selection. In particular, it represents an arc in the network topology, starting from an already queried peer and going to a destination (and still unvisited) one, and is structured as a quadruple: ($G$.from, $G$.to, $G$.relevance, $G$.active), where $G$.from is the goal's starting peer, $G$.to is the goal's destination (a neighbor of $G$.from), $G$.relevance is the relevance used for ranking and $G$.active is a boolean value which is used for $G$'s logical deletion. As we will see, inactive goals are used to reconstruct the path to an already queried peer. $GL$ is always kept ordered on the basis of the goals' relevances, which are calculated by means of the `fScore()` function (line 02) for the semantic part of the problem. In fact, as for the other algorithm, a combined routing approach is necessary, so we estimate a semantic and a multimedia relevance whose values have to be aggregate. Notice that active goals are always kept ahead inactive ones in $GL$'s ordering. Similarly to DF, the `fScore()` argument represents the semantic relevance of the goal but, in this case the computation is `compose(`$q.sim$`,combine(`$q.f, SRI$`[N]))`[4], since, differently

[4] `compose` is a fuzzy composition function. Possible choices are the minimum or the algebraic product; for more details see [8].

from DF, goals must be globally comparable and thus the whole path originating from the querying peer must be taken into account. Further, as we will see later in detail, different `fScore()` functions can be adopted in order to favor different priorities. The multimedia relevance (line 03) is estimated in the same manner as in DF technic.

Following algorithm `executeG(`$q$`,`$C$`,`$L$`,`$GL$`)`: Each Peer, after local execution (line 00), updates $GL$ with the current neighbors' goals (lines 01-05), orders $GL$ (line 06) and simply selects the next goal by accessing the top goal (line 08). As in DF, $L$ is used to avoid cycles (line 09). Then, if NextG starts from the current peer (this means that the destination peer is a neighbor) query forwarding is executed similarly to DF (lines 20-23). Instead, if this is not the case (lines 15-18), the query is sent back one hop at a time to the peer that inserted NextG in $GL$. Notice that goals are not directly removed from GL after being selected at line 08. Instead, they are set as inactive (line 22) (i.e. already visited) before forwarding the query to their destination peer (line 23). The path to reach a previously queried peer is obtained through the function `selectNextHop(`$GL$`)` (line 17), whose code will not be shown in detail for simplicity of presentation; such function exploits $GL$ inactive goals' information in order to identify, hop by hop, the route backtracking to the selected goal. The traversed peers are sent message `restoreG(`$id$`,`$L$`,`$GL$`)`. Obviously going back to potentially distant goals (peers) has a cost in terms of efficiency but always ensures the highest possible effectiveness, since the most relevant discovered peers are always selected. Notice that, in this case, backtracking is not limited to the chain of calling peers, since the selected goal does not necessarily start from one of them.

When, during backtracking, the current peer identifies that all goals starting from itself have been navigated, it can remove unnecessary information from $NS$ and $GL$. These steps are performed through function `clean(`$id$`,`$GL$`,`$NS$`)`, invoked by peer with identifier $p$.ID=id at lines 13 and 16. In particular, the function removes the navigation state related to $q$ from $NS$ and the goal $G$ having $G$.to=$p$.ID, since the query will not be forwarded again to $p$. Finally, going back to main execution, if no more goals are available in $GL$ the execution is stopped (line 10-11), while, if only inactive goals are left, final backtracking is executed (lines 12-14).

Based on the G model, we devised two routing policies, which differ on the basis of the `fScore()` function:

- **G policy**: The `fScore()` function is the identity and, thus, the goals are selected solely on the basis of their semantic and multimedia relevance;
- **GH (Global Hybrid) policy**: This "hybrid" policy chooses goals following a trade-off between effectiveness and efficiency. This is achieved by introducing an ad-hoc parameterizable `fScore()` function, which does not only consider a goal $G$'s semantic and multimedia relevance $smRel$ but also its distance $hops$ (expressed in number of hops) from the current peer: `fScore`$(smRel) = smRel/(hops)^k$, $k = 0 \ldots \infty$. By simply adjusting the value of $k$, the GH policy can be easily tuned more on efficiency ($k \to \infty$) or on effectiveness ($k \to 0$).

# 7 Rewriting

In this section, we describe how semantic mappings are used to reformulate the query as we want to obtain a rewriting using concepts from an ontology which is different from $Ont_0$. Let we now define notation used in the follow:

- $C_x \in \mathcal{C}$ is a *concept* (e.g. `company`);
- $\pi_x \in \mathcal{PR}$ is a *property* (e.g. `hasName`);
- concepts $\mathcal{C}$, together with *literals* $\mathcal{L}$ (e.g., "Fiat") and *types* $\mathcal{T}$ (e.g., `xsd:String`), are generically denoted as *nodes* $\mathcal{N}$, i.e., $\mathcal{N} = \mathcal{C} \cup \mathcal{L} \cup \mathcal{T}$;
- $r_x = (n_{x_s}, \pi_x, n_{x_o})$, $r_x \in \mathcal{R}$, is a *relationship*, where $n_{x_s}$ is the subject and $n_{x_o}$ is the object, e.g., (`company`, `hasName`, "Fiat"). Further, $s(r_x)$ and $o(r_x)$ are used alternatively to denote the subject and object of $r_x$, respectively. Finally, we will say that a node $n_x$ *is involved* in $r_x$, written $n_x \lhd r_x$, if $n_x$ is the subject or object of such relationship;
- $\overline{r}_x$ is a *path* of consecutive relationships, i.e., $\overline{r}_x = \{r_{x_1}, \ldots, r_{x_n}\}$ and $o(r_{x_1}) = s(r_{x_2}), \ldots, o(r_{x_{n-1}}) = s(r_{x_n})$. Further, $start(\overline{r}_x)$ and $end(\overline{r}_x)$ denote the starting node (i.e., $s(r_{x_1})$) and ending node (i.e., $o(r_{x_n})$) of the path, respectively. Finally, a node $n_x$ *is involved* in $\overline{r}_x$, $n_x \lhd \overline{r}_x$, if it is involved in one of its relationships, i.e., $\exists\, r_{x_i} \in \overline{r}_x : n_x \lhd r_{x_i}$.

An ontology $Ont$ is a graph $(\mathcal{N}, \mathcal{R})$, where $\mathcal{N}$ are the nodes and $\mathcal{R} \subseteq \mathcal{N} \times \mathcal{PR} \times \mathcal{N}$ are the arcs. A query $q$ in internal form is also a graph $(\mathcal{N}^q, \mathcal{R}^q)$, where $\mathcal{N}^q$ are the query nodes, $\mathcal{R}^q \subseteq \mathcal{N}^q \times \mathcal{PR}^q \times \mathcal{N}^q$ are the arcs, and $\mathcal{N}^q \subseteq \mathcal{N}$, $\mathcal{R}^q \subseteq \mathcal{R}$. As in previous sections, the set of (directional) semantic mappings is denoted as $\mathcal{M} = \{M_{i,j}\}$, where $M_{i,j}$ maps concepts of $Ont_i$ into concepts of $Ont_j$. Each mapping $M_{i,j}$ involves:

1. Correspondences between concepts: They can be one-to-one or, more generally, one-to-many. If $C$ is a concept of $Ont_i$, then: $M_{i,j}(C) = \{C'_1 \vee \ldots \vee C'_n\}$, i.e., $(C, C'_1), \ldots, (C, C'_n) \in M_{i,j}$, where $C'_k$ are concepts in $Ont_j$;
2. Correspondences between relationships: They are of the one-to-many kind and, in general, are relationship-to-path ones. If $r$ is a relationships of $Ont_i$, then: $M_{i,j}(r) = \{\overline{r}'\}$, i.e., $(r, \overline{r}') \in M_{i,j}$, where $\overline{r}'$ is a path in $Ont_j$.

Note that only concepts $\mathcal{C}$ and relationships/paths $\mathcal{R}$ are mapped by $M_{i,j}$. Literals $\mathcal{L}$ and types $\mathcal{T}$ implicitly correspond between different ontologies with an identity mapping. Further, note that, for a relationship $r$ to be in correspondence with a path $\overline{r}'$, the nodes at their extremes must also be in correspondence:

$$(r, \overline{r}') \in M_{i,j} \Rightarrow (s(r), start(\overline{r}')),\ (o(r), end(\overline{r}')) \in M_{i,j}$$

## 7.1 SPARQL grammar

Before starting to describe the rewriting algorithm it is necessary to provide a little description about the used query language SPARQL [1] and, in particular, we are interested in describing how we have modified its grammar. In fact, starting from the complete one, we have

created a personalized grammar containing only those fundamental constructions which we are interested to for our purpose. This has been done because it was not useful to implement such particular case of rewriting that will never be used.

Moreover, since our project is to be able to retrieve both semantic and multimedia information, it has been necessary to extend our grammar with a new construction: `LIKE`. It represents a clause in the `WHERE` statement and its syntax is:

$$LIKE(?var, URL)$$

as shown, for example, in Figure 1 where the LIKE condition is:

```
LIKE(?image, http://www.image.com/23.jpeg)
```

This construction syntax is very simple as it is not possible to combine different lines in a logical expression, as we can do with the `FILTER` statement, but it is necessary to write a complete condition for each of the multimedia object we need. For example:

```
WHERE  {  ?x a ns:Product.
          ?x ns:origin "Italy".
          ?x ns:image ?image.
          LIKE(?image, http://www.image.com/23.jpeg) ||
          LIKE(?image, http://www.image.com/24.jpeg)}
```

In this case we are interested in all images similar to "http://www.image.com/23.jpeg" or similar to "http://www.image.com/24.jpeg" so we have to specify two distinct `like` statements as two different `where` conditions.

We present the complete SPARQL grammar we use in Appendix A.

## 7.2 Query rewriting algorithms

Given a query $q_i = (\mathcal{N}_i^q, \mathcal{R}_i^q)$, expressed in terms of $Ont_i$, by exploiting $M_{i,j}$, $j = 1, \ldots, k$, the goal of query rewriting is to reformulate $q_i$ in terms of $Ont_j$, $j = 1, \ldots, k$. Algorithm 7.1 GRAPH REWRITING takes in input $q_i$ and $\mathcal{M}$ and produces as output a sorted list $RR$ of ranked rewritings $R_j = (rq_j, \mu_{rq_j})$, $j = 1, \ldots, k$, where $\mu_{rq_j}$ is the *score* assigned to rewriting $rq_j$. Rewriting is performed from internal (graph) form to internal form basically by unfolding. Let us now analyze the algorithm in detail.

**Algorithm 7.1** GRAPH REWRITING

**Input:** $q_i = (\mathcal{N}_i^q, \mathcal{R}_i^q)$ query to be rewritten (expressed in terms of $Ont_i$)
$\mathcal{M} = \{M_{i,j}\}$, $j = 1, \ldots, k$ set of directional semantic mappings $Ont_i \rightarrow Ont_j$
**Output:** $RR$ sorted list of ranked rewritings over source ontologies

1: **for** $j = 1$ to $k$ **do**
2: $CORR \leftarrow \emptyset$
3: $CORR \leftarrow$ CONCEPT REWRITING $(q_i, M_{i,j}, CORR)$
4: $CORR \leftarrow$ RELATIONSHIP REWRITING $(q_i, M_{i,j}, CORR)$
5: $rq_j \leftarrow$ REWRITING AND SCORE COMPUTATION $(q_i, CORR)$
6: $RR \xleftarrow{add} rq_j$
7: **end for**
8: **return** $RR$

Basically, the algorithm performs a cycle on the different source ontologies $Ont_j$, $j =$

$1, \ldots, k$ (line 1) and returns the corresponding rewriting $rq_j$. The rewriting is produced in different phases:

- First, the algorithm tries to rewrite the concepts of the original query, by calling sub-algorithm CONCEPT REWRITING (line 3). The found correspondences between the concepts are kept in a set $CORR$, $CORR \subseteq M_{i,j}$;
- Then, by exploiting the mapping and the concepts selected in the previous phase, relationships are rewritten by sub-algorithm RELATIONSHIP REWRITING (line 4), which updates set $CORR$ with the discovered correspondences between relationships or between a relationship and a path;
- Finally, query $q_j$ in graph form is produced and is assigned a score, forming the final rewriting $rq_j$ (line 6, sub-algorithm REWRITING AND SCORE COMPUTATION).

Now we provide a brief description of the three sub-algorithms which code is completely shown in appendix B.

Algorithms B.1 CONCEPT REWRITING and B.2 RELATIONSHIP REWRITING have completely analogues way to act. Both of them takes as input the query to be rewritten, the semantic mapping between the two involved ontologies and the set of previously found correspondences for the current rewriting (in the case of the first algorithm it is empty). Then both algorithms search for, respectively, concept and relationship correspondences into the mapping. In detail algorithm B.1 CONCEPT REWRITING cycles through the concepts $\mathcal{C}_i^q$ of $q_i$ (line 1) and searches for correspondences in the mapping; if found they are stored in $CORR$ (lines 6-8). Note that for concepts with one-to-many mappings in $M_{i,j}$ all possible correspondences are stored while if no correspondences are available for a concept $C_x$ rewriting is not possible and the algorithm is stopped (line 4). Algorithm B.2 RELATIONSHIP REWRITING cycles through the relationships $\mathcal{R}_i^q$ of $q_i$ (line 1) then, in lines 6-8, $M_{i,j}$ is searched for correspondences: if available, they are added to $CORR$ (line 7), otherwise for the relation $r_x$ rewriting is not possible and the algorithm is stopped (line 4). Note that in line 6 and 7 $\overline{r}'$ is used as, in this phase, both relationship-to-relationship and relationship-to-path correspondences are added to the final structure $CORR$ if they exist.

In the last phase, Algorithm B.3 REWRITING AND SCORE COMPUTATION builds graph $rq_j$ by extracting concepts (lines 2-4) and relationships (lines 5-12) from $CORR$. Types and literals are kept unchanged (lines 13-14). The value $\mu_{rq_j}$ represents a global score associated to the rewritten query and it expresses the accuracy, that is the semantic similarity, in respect to the original query. As we can see in lines 3, 7 and 10 this score is calculated as combination of membership grades given by involved concept mappings. Since we assume a conjunction query, membership grades combination is calculated with the fuzzy conjunction. Moreover, if we have multiple mappings and so the query is rewritten as union of more queries, its score is obtained with the fuzzy disjunction between those query scores.

## 7.3 Rewriting output

The rewriting module provides as output an XML file with obtained queries and some additional information about the rewriting procedure. A such file example is shown in Figure 14. The first section of the file specifies the rewriting process input: the original query and the name of the peer that query is posed on. Instead the second section is about the output as

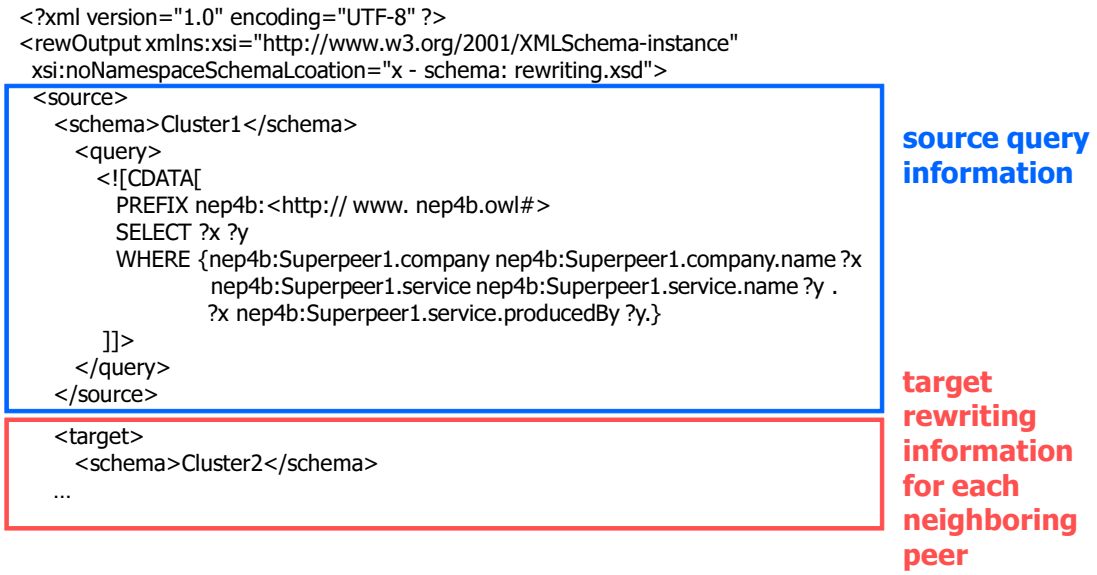



Figure 14: Rewriting module output

it specifies the name of the schema in respect the query is rewritten to, the global score and the rewritten query set.

In Figure 15 we have detailed information provided for each rewritten query: the query itself, the score expressing the query accuracy and the ”rewritingPerc” which give us the percentage of rewritten elements. Moreover it follows a list of query concepts indicating for each of them if it has been rewritten or not.

## 7.4 Examples

**Example 7.1** *Let's start with a very simple example based on Figure 16(a). This is the situation whose output file is shown in Figures 14 and 15. In particular we are interested in knowing all those companies which produce services. The query posed to Superpeer1 is so this one:*

```
PREFIX nep4b:<http://www.nep4b.owl#>
SELECT ?x ?y
WHERE {nep4b:Superpeer1.company nep4b:Superpeer1.company.name ?x .
       nep4b:Superpeer1.service nep4b:Superpeer1.service.name ?y.
       ?x nep4b:Superpeer1.service.producedBy ?y.}
```

*Superpeer1 concepts involved in this query are:* `company`, `company.name`, `service`, `service.name` *and* `service.producedBy`. *In order to query Superpeer2 we need to extract from the mapping between those two schemas, correspondences for such concepts. In this very simple case each of the Superpeer1 elements can be mapped into a Superpeer2 element. In particular we obtain:*

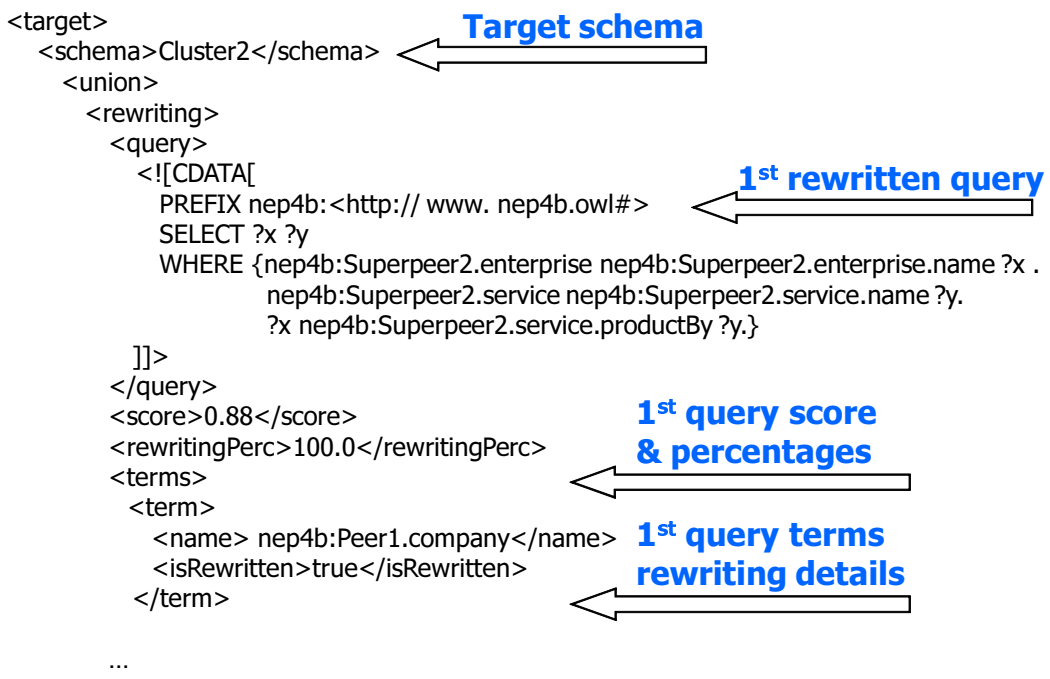



Figure 15: Focus on the output XML file portion relative to a single query

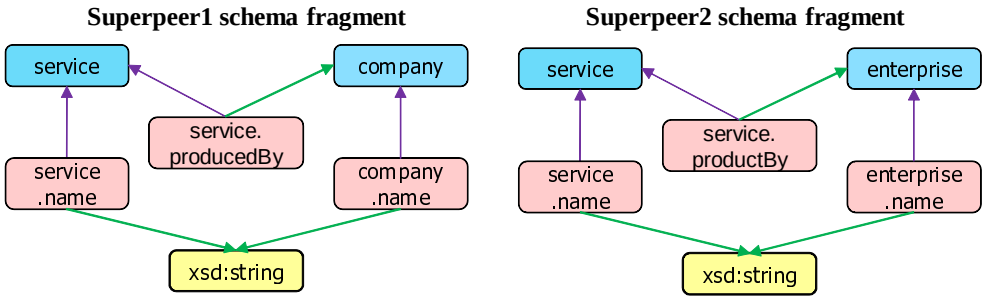
$$
\begin{aligned}
\textit{Superpeer1.company} &\longrightarrow \textit{Superpeer2.enterprise}\\
\textit{Superpeer1.service} &\longrightarrow \textit{Superpeer2.service}\\
\textit{Superpeer1.company.name} &\longrightarrow \textit{Superpeer2.enterprise.name}\\
\textit{Superpeer1.service.name} &\longrightarrow \textit{Superpeer2.service.name}\\
\textit{Superpeer1.service.producedBy} &\longrightarrow \textit{Superpeer2.service.productBy}
\end{aligned}
$$

*so the rewritten query is simply a substitution of concepts:*

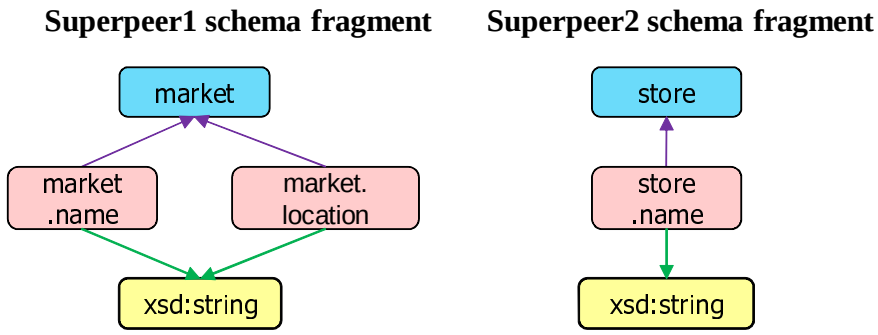
```
PREFIX nep4b:<http://www.nep4b.owl#>
SELECT ?x ?y
WHERE {nep4b:Superpeer2.enterprise
       nep4b:Superpeer2.enterprise.name ?x .
      nep4b:Superpeer2.service nep4b:Superpeer2.service.name ?y .
      ?x nep4b:Superpeer2.service.productBy ?y.}
```

*As the simplicity of this case it is reasonable to await good results in term of rewriting grades. In fact, the rewriting percentage is of 100% and the score expressing the global query accuracy is 0.88 as all the mapping concepts have a good similarity value.*

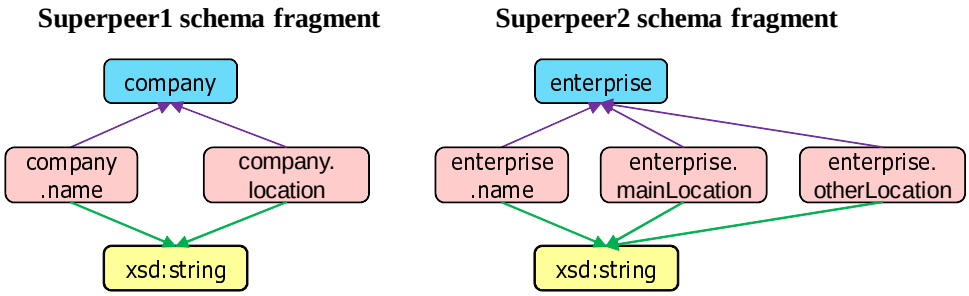
**Example 7.2 (`Non-mapped concept`)** *Let's refer to Figure 16(b). We are interested in knowing the name and the location of all markets in 'Modena' and 'Bologna' excluding those ones whose name is 'Bigstore'. The query posed to Superpeer1 is:*




(a) Example 7.1




(b) Example 7.2




(c) Example 7.3

Figure 16: Different situation may be find during query rewriting

```
PREFIX nep4b:<http:// www.  nep4b.owl#>
SELECT ?x ?y
WHERE {nep4b:Superpeer1.market nep4b:Superpeer1.market.name ?x .
       nep4b:Superpeer1.market nep4b:Superpeer1.market.location ?y .
FILTER (((?y = 'MODENA') OR (?y = 'BOLOGNA') ) AND
        (?x = 'BIGSTORE'))}
```

*When we rewrite the query for Superpeer2 we chance on a non-mapped concept. In fact we have:*

*Superpeer1.market* $\longrightarrow$ *Superpeer2.store*
*Superpeer1.market.name* $\longrightarrow$ *Superpeer2.store.name*

*but there isn't any mapping correspondence for* `market.location` *in target schema. For this reason, the rewriting procedure does not only consist of concept substitutions but it is more complex. In particular, it is necessary to eliminate triples involving unmatched elements. In this case a filter condition is also present, so it must be unfolded in order to keep only significant conditions. This is not a light process because, as here, such conditions are grafted in a logical expression. The resulting query is:*

```
PREFIX nep4b:<http:// www.  nep4b.owl#>
SELECT ?x
WHERE {nep4b:Superpeer2.store nep4b:Superpeer2.store.name ?x .
FILTER (?x = 'BIGSTORE') }
```

*Naturally rewriting percentage is smaller than example 7.1 and in this case it is 66% as 2 of 3 concepts of Superpeer1 have been mapped into concepts of Superpeer2.*

**Example 7.3 (`Multi-mapped concept`)** *Have a look at Figure 16(c); we are interested in findings all the companies, excluding those in 'Bologna' and showing their name and their location. For the Superpeer1 the query is:*

```
PREFIX  nep4b:<http:// www. nep4b.owl#>
SELECT  ?x ?y
WHERE { nep4b:Superpeer1.company nep4b:Superpeer1.company.name ?x .
        nep4b:Superpeer1.company nep4b:Superpeer1.company.location ?y .
FILTER  (?y= 'BOLOGNA') }
```

*Analyzing the mapping file we find that:*

*Superpeer1.company* $\longrightarrow$ *Superpeer2.enterprise*
*Superpeer1.company.name* $\longrightarrow$ *Superpeer2.enterprise.name*
*Superpeer1.company.location* $\longrightarrow$ *Superpeer2.enterprise.mainLocation*
*Superpeer1.company.location* $\longrightarrow$ *Superpeer2.enterprise.otherLocation*

*In this case, all the Superpeer1 concepts are mapped into Superpeer2 concepts and, moreover,* `company.location` *is present in two different matching. We know that both couples are right at the same level because we have defined the matching algorithm to allow two right match if their similarity difference is smaller than a given threshold. In order to extract all relevant results in this case, it is necessary to perform two different queries, one relative to the map pair (*`company.location`*,* `enterprise.mainLocation`*) and one relative to (*`company.location`*,* `enterprise.otherLocation`*). The obtained queries are:*

```
PREFIX nep4b:<http:// www.  nep4b.owl#>
SELECT ?x ?y
```

```
WHERE {nep4b:Superpeer2.enterprise
        nep4b:Superpeer2.enterprise.name ?x .
       nep4b:Superpeer2.enterprise
        nep4b:Superpeer2.enterprise.mainLocation ?y .
FILTER (?y = 'BOLOGNA') }

PREFIX nep4b:<http:// www.  nep4b.owl#>
SELECT ?x ?y
WHERE {nep4b:Superpeer2.enterprise
        nep4b:Superpeer2.enterprise.name ?x .
       nep4b:Superpeer2.enterprise
        nep4b:Superpeer2.enterprise.otherLocation ?y .
FILTER (?y = 'BOLOGNA') }
```

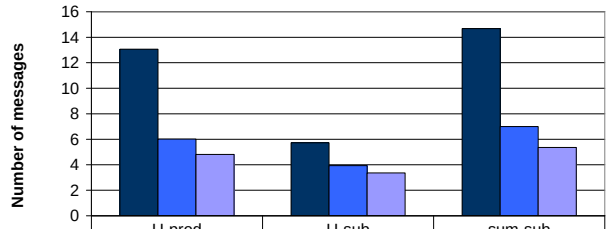
*After that results must be grouped together with a simple union operation and for this reason the global score is obtained as the fuzzy disjunction between the global scores of the two single queries.*

# 8 Experiments

In this section we present different experiments on routing mechanism and in particular the first one is about the lonely SRI technic (see section 8.1), the second one is performed on combined routing (section 8.2) and the third one on different routing policies (section 8.3).

For all of these we used a *simulation environment by which we were able to reproduce the main conditions characterizing a PDMS semantic network without using a real P2P system. Further, we could easily abstract from network and communication issues, while maintaining full control on the internal dynamics of the network management. The simulation module is based on SimJava 2.0, a discrete, event-based, general purpose simulator. Through this framework we modeled scenarios corresponding to networks of semantic peers, each with its own schema describing a particular reality. We chose peers belonging to different semantic categories, where the schemas of the peers in the same category describe the same topic from different points of view. As in [14], 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 structural and terminological variations on the original ones. Then, we distributed these schemas in the network in a clustered way, i.e. the schemas belonging to the same semantic category are located close to each other. This reflects realistic scenarios where nodes with semantically similar content are often clustered together. As to the topology of the semantic network of mappings connecting the peers, we tested our techniques on different alternatives generated with the BRITE topology generator tool. The mean size of our networks was in the order of some hundreds of nodes.*

*In order to evaluate the effectiveness and efficiency of routing mechanisms, we simulated the querying process by instantiating different queries on randomly selected peers and propagating them until a stopping condition is reached. We considered two alternatives: stopping the querying process when a given number of hops (*hops*) has been performed and measuring the quality of the results (*satisfaction *or* combined satisfaction*) or, in a dual way, stopping when a given satisfaction is obtained and measuring the required number of hops. Satisfaction is a specifically introduced quantity that grows proportionally to the goodness of the results returned by each queried peer.*




| | U-prod | U-sub | sum-sub |
|---|---|---|---|
| Random | 13.06 | 5.73 | 14.68 |
| SemMap | 6.02 | 3.93 | 7 |
| SRI | 4.81 | 3.36 | 5.36 |

Figure 17: Selection of functions

## 8.1 SRIs Experiments

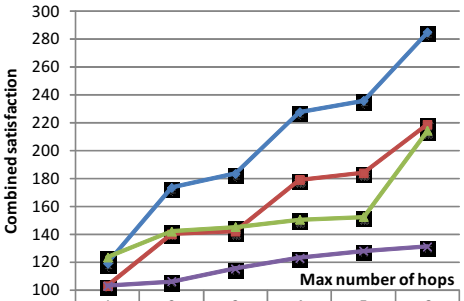
*In this section, we compare our neighbor selection mechanism based on SRIs (*SRI*) with a mechanism where the selection of the next neighbor to be visited is based on the semantic mapping values (*SemMap*) and with a baseline corresponding to a random strategy (*Random*). Notice that all the results we present are computed as a mean on several hundreds query executions.*

*We start by considering tree topologies and a satisfaction goal as stop condition. Our first aim is to evaluate if the information provided by the SRIs, built using different combinations of the proposed functions for aggregation and composition, is useful to perform a good routing mechanism. In Figure 17 we show three scenarios describing the behavior of the routing mechanism exploiting the Random, SemMap and SRI neighbor selection. Besides the ones presented in the previous sections, we also considered the sum for aggregation and a variant of the difference for composition (sub in the figure). The three scenarios differ for the choice of functions and the satisfaction goal, but in each of them we have similar results: The SemMap strategy is by far better than the Random strategy, but the SRI one requires even fewer messages to reach the goal. In particular, the first scenario involving the $U$ and the product functions shows the higher number of saved messages (from 6.02 to 4.81, which leads to an improvement of 25%); thus, for the following tests we will refer to this pair of functions.*

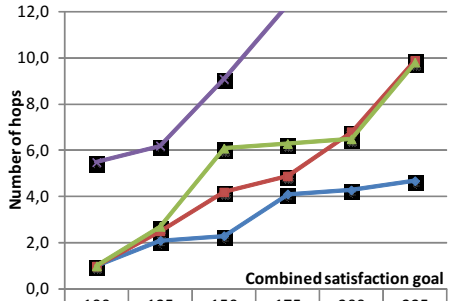
## 8.2 Combined routing experiments

*In this section we present an initial set of experiments we performed in order to evaluate our combined query routing approach. Notice that, since we are currently in the initial phase of our testing, the considered scenarios are not particularly complex; in the future we will enrich them with more complicated and larger ones. For our experiments, we exploited our simulation environments for putting into action the SRI [7] and MRoute [4] approaches. Through these environments we modeled scenarios corresponding to networks of semantic peers, each with its own schema, consisting of a small number of concepts, and a repository of multimedia objects. As said before, we chose peers belonging to different semantic categories, where the peers in the same category have schemas describing the same topic from different points of view but now we considered also multimedia data related to that topic. As to the multimedia contents, we use 1300 images taken from the Corel Photo CDs and characterized by two MPEG-7 standard features: scalable color and edge histogram. Each contribution to the combined satisfaction is computed by combining the semantic mapping scores of the traversed peers (satisfaction measure [9]) and the multimedia similarity scores of the retrieved objects. The search strategy employed is the depth first search (DFS). In our experiments we compare our neighbor selection mechanism based on a combination of SRIs and MRoute (Comb) with the two mechanisms which only exploit the SRI (SRI) and MRI (MRI) values and with a baseline corresponding to a random strategy (Rand). The employed aggregation function is the mean. Notice that all the results we present are computed as a mean on some query executions.*

*Figure 18 shows the trend of the obtained combined satisfaction when we gradually vary the stopping condition on hops (left) and the dual situation (right) where the number of hops required to reach a given satisfaction goal is measured. As we expected, both the SRI and the MRI strategies outperform the Rand one, but, as we can see, the winner is the Comb*

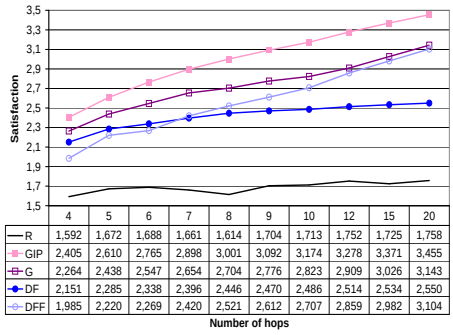



| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Comb | 119,42 | 173,61 | 183,61 | 227,85 | 235,85 | 284,94 |
| MRI | 103,17 | 140,23 | 142,23 | 179,10 | 184,10 | 218,98 |
| SRI | 123,67 | 142,23 | 145,33 | 150,48 | 152,48 | 214,55 |
| Rand | 103,17 | 106,34 | 115,54 | 123,10 | 128,10 | 131,24 |



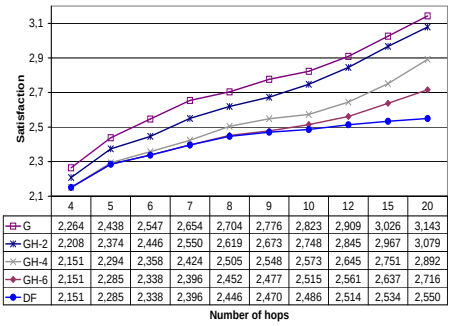

| | 100 | 125 | 150 | 175 | 200 | 225 |
|---|---|---|---|---|---|---|
| Comb | 1,0 | 2,1 | 2,3 | 4,1 | 4,3 | 4,7 |
| MRI | 1,0 | 2,5 | 4,2 | 4,9 | 6,8 | 9,9 |
| SRI | 1,0 | 2,7 | 6,1 | 6,3 | 6,5 | 9,8 |
| Rand | 5,5 | 6,2 | 9,1 | 12,3 | 15,2 | 17,7 |

Figure 18: Obtained combined satisfaction for a given number of hops (left) and mean number of hops for a combined satisfaction goal (right)




| | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 12 | 15 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|
| R | 1,592 | 1,672 | 1,688 | 1,661 | 1,614 | 1,704 | 1,713 | 1,752 | 1,725 | 1,758 |
| GIP | 2,405 | 2,610 | 2,765 | 2,898 | 3,001 | 3,092 | 3,174 | 3,278 | 3,371 | 3,455 |
| G | 2,264 | 2,438 | 2,547 | 2,654 | 2,704 | 2,776 | 2,823 | 2,909 | 3,026 | 3,143 |
| DF | 2,151 | 2,285 | 2,338 | 2,396 | 2,446 | 2,470 | 2,486 | 2,514 | 2,534 | 2,550 |
| DFF | 1,985 | 2,220 | 2,269 | 2,420 | 2,521 | 2,612 | 2,707 | 2,859 | 2,982 | 3,104 |

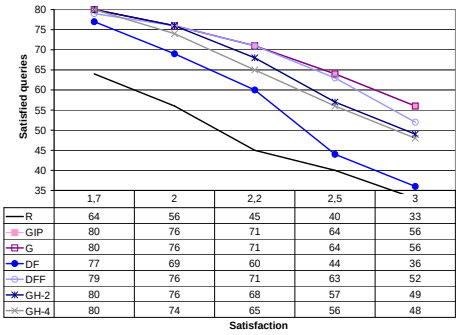
Figure 19: Satisfaction reached by routing policies given a maximum number of hops




| | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 12 | 15 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|
| G | 2,264 | 2,438 | 2,547 | 2,654 | 2,704 | 2,776 | 2,823 | 2,909 | 3,026 | 3,143 |
| GH-2 | 2,208 | 2,374 | 2,446 | 2,550 | 2,619 | 2,673 | 2,748 | 2,845 | 2,967 | 3,079 |
| GH-4 | 2,151 | 2,294 | 2,358 | 2,424 | 2,505 | 2,548 | 2,573 | 2,645 | 2,751 | 2,892 |
| GH-6 | 2,151 | 2,285 | 2,338 | 2,396 | 2,452 | 2,477 | 2,515 | 2,561 | 2,637 | 2,716 |
| DF | 2,151 | 2,285 | 2,338 | 2,396 | 2,446 | 2,470 | 2,486 | 2,514 | 2,534 | 2,550 |

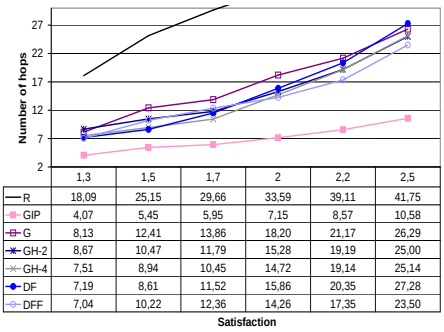
Figure 20: Comparison of GH policy varying $k$ w.r.t. G and DF policies

*mechanism. In particular, the difference between SRI - MRI and Comb performance appears closer in the initial part of the graphs but becomes increasingly more significant at growing stop conditions. This means that Comb is indeed able to discriminate better subnetworks to explore and consequently increases the combined satisfaction and decreases the number of hops in a more substantial way. As an example of this behavior, when we executed a query involving the concept Monument and a similarity constraint on an image of the Pisa tower, we observed that the Rand strategy worked by randomly selecting peers which were completed unrelated with the image and the concept required. On the other hand, the SRI strategy proceeded by firstly selecting some peers which have the concept Monument (and thus a very high SRI's score) but no image similar to the Pisa tower. Further, the MRI approach preferred some peers which store the images of some chimneys (whose multimedia features were very similar to the Pisa tower's ones) even if they were associated to the concept Factory. Only the Comb strategy was able to identify the best peers, i.e. the peers where the images of the Pisa tower are associated to concepts similar to the required one.*




| | 1,7 | 2 | 2,2 | 2,5 | 3 |
|---|---|---|---|---|---|
| R | 64 | 56 | 45 | 40 | 33 |
| GIP | 80 | 76 | 71 | 64 | 56 |
| G | 80 | 76 | 71 | 64 | 56 |
| DF | 77 | 69 | 60 | 44 | 36 |
| DFF | 79 | 76 | 71 | 63 | 52 |
| GH-2 | 80 | 76 | 68 | 57 | 49 |
| GH-4 | 80 | 74 | 65 | 56 | 48 |

Figure 21: Percentage of queries that reach a given satisfaction



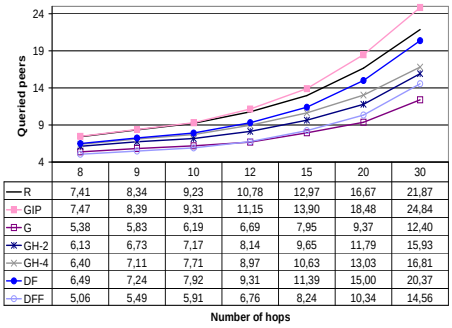

| | 1,3 | 1,5 | 1,7 | 2 | 2,2 | 2,5 |
|---|---|---|---|---|---|---|
| R | 18,09 | 25,15 | 29,66 | 33,59 | 39,11 | 41,75 |
| GIP | 4,07 | 5,45 | 5,95 | 7,15 | 8,57 | 10,58 |
| G | 8,13 | 12,41 | 13,86 | 18,20 | 21,17 | 26,29 |
| GH-2 | 8,67 | 10,47 | 11,79 | 15,28 | 19,19 | 25,00 |
| GH-4 | 7,51 | 8,94 | 10,45 | 14,72 | 19,14 | 25,14 |
| DF | 7,19 | 8,61 | 11,52 | 15,86 | 20,35 | 27,28 |
| DFF | 7,04 | 10,22 | 12,36 | 14,26 | 17,35 | 23,50 |

Figure 22: Mean number of hops needed to reach a given satisfaction

## 8.3 Routing policies experiments

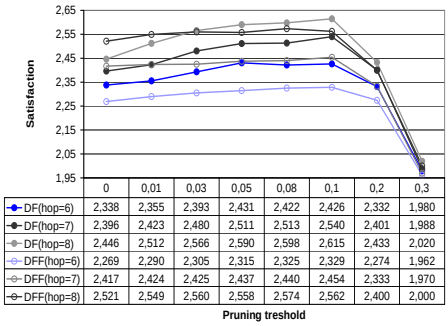
*In this section, we compare our routing strategies presented in Section 6 considering two further policies: The Random (R) one, which moves randomly in the network and corresponds to a baseline, and the Global IP-based (GIP) one. The GIP strategy is a variation of the Global (G) mechanism: A direct connection is established between the current peer and the peer chosen for the following step, avoiding the hops needed to reach it in the original network topology. This policy involves the modification of the PDMS network (the IP addresses of the visited peers are stored and new connections are created when necessary) and it can not be considered a real P2P strategy, but it is an interesting upper-bound to be shown. Notice that all the results we present are computed as a mean on several hundreds of query executions on several networks.*

*We start by studying the effectiveness of the routing strategies with two types of experiments. For the first one, Figure 19 shows the trend of the obtained satisfaction when we gradually vary the stopping condition on hops. As we expected, the Random strategy is outdistanced by the others and the GIP one represents the upper-bound, since all the available hops are exploited for querying peers (there are no backtracking hops). Among the others, the G policy shows the best behavior as it selects for each step the available peer with the higher sim value. Further, the DFF mechanism is initially less effective than the DF one, since it uses a large number of hops for performing its "fan" exploration. Nevertheless, for higher stopping conditions, it becomes increasingly more effective until it outperforms the DF policy and almost reaches the G one: This is due to the fact that it visits nearer peers, which have a higher probability to provide better results. The results for the GH policy are shown in Figure 20, where the trends for different values of the $k$ parameter are represented. Notice that all the curves for the GH strategy are included in the area delimited between the G and the DF ones: The G and DF policies are indeed two particular cases of the GH one, where only the relevance (for G) or hops (for DF) component is considered, i.e. $k$ in the formula is set to $0$ (for G) or $\infty$ (for DF). Further, increasing the $k$ value, the obtained behavior gets closer to the G one, as more importance is given to the relevance component. The results for the second type of experiments are represented in Figure 21, where the percentage of satisfied queries (i.e the queries for which the level of satisfaction given as stopping condition is reached) is shown for different levels of satisfaction. The G policy confirms to be the most effective one, while the performances of the GH ones decrease with the value of $k$. As for the family of the DF strategies, the earlier explained difference between DF and DFF is even further evidenced.*




| | 8 | 9 | 10 | 12 | 15 | 20 | 30 |
|---|---|---|---|---|---|---|---|
| R | 7,41 | 8,34 | 9,23 | 10,78 | 12,97 | 16,67 | 21,87 |
| GIP | 7,47 | 8,39 | 9,31 | 11,15 | 13,90 | 18,48 | 24,84 |
| G | 5,38 | 5,83 | 6,19 | 6,69 | 7,95 | 9,37 | 12,40 |
| GH-2 | 6,13 | 6,73 | 7,17 | 8,14 | 9,65 | 11,79 | 15,93 |
| GH-4 | 6,40 | 7,11 | 7,71 | 8,97 | 10,63 | 13,03 | 16,81 |
| DF | 6,49 | 7,24 | 7,92 | 9,31 | 11,39 | 15,00 | 20,37 |
| DFF | 5,06 | 5,49 | 5,91 | 6,76 | 8,24 | 10,34 | 14,56 |

Figure 23: Mean number of queried peers given a maximum number of hops




| | 0 | 0,01 | 0,03 | 0,05 | 0,08 | 0,1 | 0,2 | 0,3 |
|---|---|---|---|---|---|---|---|---|
| DF(hop=6) | 2,338 | 2,355 | 2,393 | 2,431 | 2,422 | 2,426 | 2,332 | 1,980 |
| DF(hop=7) | 2,396 | 2,423 | 2,480 | 2,511 | 2,513 | 2,540 | 2,401 | 1,988 |
| DF(hop=8) | 2,446 | 2,512 | 2,566 | 2,590 | 2,598 | 2,615 | 2,433 | 2,020 |
| DFF(hop=6) | 2,269 | 2,290 | 2,305 | 2,315 | 2,325 | 2,329 | 2,274 | 1,962 |
| DFF(hop=7) | 2,417 | 2,424 | 2,425 | 2,437 | 2,440 | 2,454 | 2,333 | 1,970 |
| DFF(hop=8) | 2,521 | 2,549 | 2,560 | 2,558 | 2,574 | 2,562 | 2,400 | 2,000 |

Figure 24: Pruning threshold effect on query satisfaction w.r.t. DF and DFF policies

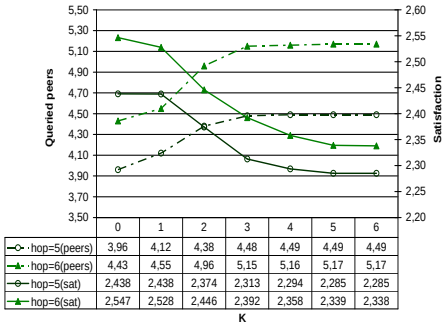
*The results of the experiments aiming at verifying the efficiency of the routing strategies are shown in Figures 22 and 23. Figure 22 is the dual perspective of the situation in Figure 19, since it represents the trend of the number of required hops for a given satisfaction goal. As in the previous graph, the R and GIP strategies act as the lower and upper bounds respectively. The G policy is instead the less efficient one, since at each step it chooses the next peer to visit without considering the number of hops needed to reach it. As for the GH strategies, obviously we have that the higher is $k$ the more efficient is the routing policy. It is interesting to note that also from this efficiency perspective, after the initial phase, the DFF policy outperforms the DF one.*

*Figure 23 shows another measure of the efficiency corresponding to the number of queried peers for a given number of hops: Given a limit of hops, we have in fact that the more efficient is a policy the lower is the number of useless hops executed and, thus, the higher is the number of queried peers. Also in this case the less efficient policy is the G one, while for the GH family higher values of $k$ led to a higher efficiency. The bad performances of the DFF strategy is due to the fact that, during "fan" explorations, it uses two hops for each queried peer (one for reaching it and one for going back), whereas the DF is the best policy (apart from the bound cases of R and GIP mechanisms) because each hop is used for querying the neighbor with the highest sim value.*

*Figure 24 explores the behavior of DF and DFF when a pruning mechanism involving a non-null $q.t$ threshold is activated. The graph shows, for both routing strategies, the satisfaction reached for a given hop limit when we vary the pruning threshold. Notice that when we use a zero threshold, and consequently apply no pruning mechanism, we obtain the same results presented earlier in the section. As can be seen, increasing the threshold leads initially to an improvement of satisfaction for both strategies. However, further increases of the threshold lead to lower values of satisfaction, signifying that useful subnetworks are pruned. The value of the optimal pruning threshold obviously depends on many factors, however we found out that setting it to $0.1$ was a good choice in most tested settings, as Figure 24 shows.*

*The graph represented in Figure 25 deepens the study of the behavior of the GH strategies, showing the trend of satisfaction (continuous lines) and queried peers (dotted lines) for a given number of hops when we vary the value of $k$. As we expected, the higher the value of $k$, the lower is the obtained satisfaction and the higher is the number of queried peers. Increasing $k$ we in fact give more importance to the hops component in the formula and consequently obtain more efficient but less effective strategies.*

*Finally, Figure 26 shows the relation between the number of queried peers and the sat-*




| | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| hop=5(peers) | 3,96 | 4,12 | 4,38 | 4,48 | 4,49 | 4,49 | 4,49 |
| hop=6(peers) | 4,43 | 4,55 | 4,96 | 5,15 | 5,16 | 5,17 | 5,17 |
| hop=5(sat) | 2,438 | 2,438 | 2,374 | 2,313 | 2,294 | 2,285 | 2,285 |
| hop=6(sat) | 2,547 | 2,528 | 2,446 | 2,392 | 2,358 | 2,339 | 2,338 |

Figure 25: $k$ influence on efficiency and effectiveness in GH policy

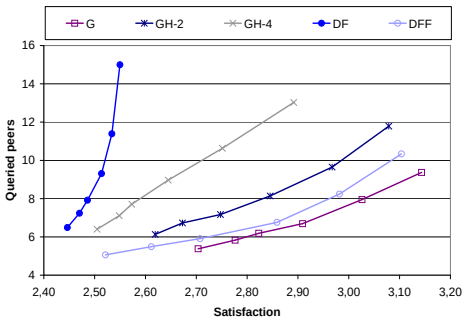



Figure 26: Effectiveness vs. efficiency of routing policies

*isfaction that every policy reaches given a maximum number of hops. In this way, we are able to visualize how the results obtained by the different policies position themselves in a combined effectiveness/efficiency plane. As expected, we observe that the G policy is the most effective one, since its curve is located near to the satisfaction axis. In contrast, the DF policy appears as the most efficient one. Moreover, we can see the effect of $k$ in the GH policy: The increasing of $k$ makes the GH policy more efficient, but less effective. Finally, notice that the DFF policy can reach satisfaction goals similar to the ones reached by the G strategy, but in a more efficient way.*

# 9 Conclusion

*This report proposed an innovative approach for processing queries effectively and efficiently in a distributed and heterogeneous environment of the NeP4B project.*

*We described two routing technics based on semantic and multimedia query routing and proposed to combine these two approaches in order to design an innovative mechanism which exploits the two main aspects characterizing the querying process in such a context: the semantics of the concepts in the peers' ontologies and the multimedia contents in the peers' repositories.*

*The query processing model proposed is founded on a fuzzy framework which provides a formal settlement for the integration of query routing techniques for both semantic and multimedia data.*

*As far as we know, this is the first research proposal specifically devised to enhance the processing of queries in a network of semantic peers which share both semantic and multimedia data.*

# A Sparql Grammar

*Input ::= Prolog SelectCause <EOF>*
*Prolog ::= ( Basedow )? ( PrefixDecl )**
*Basedow ::= "BASE" QuotedURIref*
*PrefixDecl ::= "PREFIX" <QNAME_NS> QuotedURIref*
*SelectCause ::= "SELECT" ( ( Var )+ | "*" ) ( WhereClause )? ( LimitClause )?*
*WhereClause ::= ( "WHERE" )? GroupGraphPattern*
*LimitClause ::= "LIMIT" <INTEGER_10>*
*GroupGraphPattern ::= "{" Granulaters "}"*
*Granulaters ::= FilteredBasicGraphPattern ( OptionalGraphPattern )**
*FilteredBasicGraphPattern ::= (BlockOfTriples)? (Constraint ( "." )? FilteredBasicGraphPattern)? (MultimediaConstraint ( "." )? FilteredBasicGraphPattern)**
*OptionalGraphPattern ::= "OPTIONAL" "(" FilteredBasicGraphPattern ")"*
*Constraint ::= "FILTER" "(" Expression ")"*
*MultimediaConstraint ::= "LIKE" "(" MultimediaExpression ")"*
*MultimediaExpression ::= Var "," URI*
*BlockOfTriples ::= TriplesSameSubject ( "." ( TriplesSameSubject )? )**
*TriplesSameSubject ::= VarOrURI PropertyListNotEmpty*
*VarOrTerm ::= Var | GraphTerm*
*Var ::= <VAR>*
*GraphTerm ::= URI | RDFLiteral | NumericLiteral | BooleanLiteral*
*PropertyListNotEmpty ::= Verb ObjectList ( ";" PropertyList )?*
*Verb ::= URI | "a"*
*ObjectList ::= VarOrTerm ( "," ObjectList )?*
*PropertyList ::= ( PropertyListNotEmpty )?*
*VarOrURI ::= Var | URI*
*Expression ::= ConditionalOrExpression*
*ConditionalOrExpression ::= ConditionalAndExpression ( "||" ConditionalAndExpression )**
*ConditionalAndExpression ::= ValueLogical ( "&&" ValueLogical )**
*ValueLogical ::= RelationalExpression*
*RelationalExpression ::= NumericExpression ( OptionalRelationalExpression )?*
*OptionalRelationalExpression ::= NumericExpressionOp NumericExpression*
*NumericExpressionOp ::= "=" | "!=" | "<" | "<=" | ">" | ">="*
*NumericExpression ::= AdditiveExpression*
*AdditiveExpression ::= MultiplicativeExpression ( OptionalAdditiveExpression )**
*OptionalAdditiveExpression ::= "+" MultiplicativeExpression | "-" MultiplicativeExpression*
*MultiplicativeExpression ::= UnaryExpression ( OptionalMultiplicativeExpression )**
*OptionalMultiplicativeExpression ::= "*" UnaryExpression | "/" UnaryExpression*
*UnaryExpression ::= "!" PrimaryExpression | "+" PrimaryExpression | "-" PrimaryExpression | PrimaryExpression*
*PrimaryExpression ::= Var | NumericLiteral | BooleanLiteral | ( <STRING_LITERAL1> | <STRING_LITERAL2> ) | "(" Expression ")"*
*NumericLiteral ::= <INTEGER_10> | FloatingPoint*
*RDFLiteral ::= (<STRING_LITERAL1> | <STRING_LITERAL2>) (<LANGTAG> | "b URI)?*
*BooleanLiteral ::= "TRUE" | "FALSE"*
*URI ::= QuotedURIref | QName*

*QName ::= <QNAME> | <QNAME_NS>*
*QuotedURIref ::= <Q_URIref>*
*FloatingPoint ::= <FLOATING_POINT>*

# B Rewriting Algorithms

**Algorithm B.1** CONCEPT REWRITING

**Input:** $q_i = (\mathcal{N}_i^q, \mathcal{R}_i^q)$ query to be rewritten (expressed in terms of $Ont_i$)
$M_{i,j}$ semantic mappings $Ont_i \rightarrow Ont_j$
$CORR$ set of previously found correspondences for current rewriting
**Output:** $CORR$ updated set of correspondences for current rewriting

1: **for all** $C_x \in \mathcal{C}_i^q$ **do**
2: search for concept correspondences in $M_{i,j}$
3: **if** $\nexists\, C'_x \in \mathcal{C}_j : (C_x, C'_x) \in M_{i,j}$ **then** {*no correspondences for* $C_x$}
4: rewriting is not possible
5: **else**
6: **for all** $C'_{x_k} \in \mathcal{C}_j : (C_x, C'_{x_k}) \in M_{i,j}, k = 1, \ldots, n$ **do**
7: $CORR \xleftarrow{add} (C_x, C'_{x_k})$
8: **end for**
9: **end if**
10: **end for**
11: **return** $CORR$

**Algorithm B.2** RELATIONSHIP REWRITING

**Input:** $q_i = (\mathcal{N}_i^q, \mathcal{R}_i^q)$ query to be rewritten (expressed in terms of $Ont_i$)
$M_{i,j}$ semantic mappings $Ont_i \rightarrow Ont_j$
$CORR$ set of previously found correspondences for current rewriting
**Output:** $CORR$ updated set of correspondences for current rewriting

1: **for all** $r_x = (n_{x_s}, \pi_x, n_{x_o}) \in \mathcal{R}_i^q$ **do**
2: search for relationship correspondences in $M_{i,j}$
3: **if** $\nexists\, \overline{r}'_x \subseteq \mathcal{R}_j : (r_x, \overline{r}'_x) \in M_{i,j}$ **then**
4: rewriting is not possible
5: **else**
6: **for all** $\overline{r}'_{x_k} \subseteq \mathcal{R}_j : (r_x, \overline{r}'_{x_k}) \in M_{i,j}$ **do**
7: $CORR \xleftarrow{add} (r_x, \overline{r}'_{x_k}))$
8: **end for**
9: **end if**
10: **end for**
11: **return** $CORR$

**Algorithm B.3** REWRITING AND SCORE COMPUTATION

**Input:** $q_i = (\mathcal{N}_i^q, \mathcal{R}_i^q)$ query to be rewritten (expressed in terms of $Ont_i$)
$CORR$ set of previously found correspondences for current rewriting
**Output:** $R_j = \{rq_j, \mu_{rq_j}\}$ rewriting of $q_i$ in terms of $Ont_j$, where $rq_j = (\mathcal{N}_j^q, \mathcal{R}_j^q)$

$\mathcal{N}_j^q, \mathcal{R}_j^q \leftarrow \emptyset, \quad \mu_{rq_j} \leftarrow -$
**for all** $C'_x \in \mathcal{C}_j : (C_x, C'_x) \in CORR$ **do** {*extract concepts*}
  $\mathcal{N}_j^q \xleftarrow{add} C'_x, \quad \mu_{rq_j} \leftarrow \mu_{rq_j} \circ \mu(C_x, C'_x)$
**end for**
**for all** $\overline{r}'_x \subseteq \mathcal{R}_j : (\overline{r}_x, \overline{r}'_x) \in CORR$ **do** {*extract relationships*}
  **for all** $C'_x \in \mathcal{C}_j : C'_x \lhd \overline{r}'_x$ **do**
    $\mathcal{N}_j^q \xleftarrow{add} C_x, \quad \mu_{rq_j} \leftarrow \mu_{rq_j} \circ \mu(C_x, C'_x)$
  **end for**
  **for all** $r'_x \in \overline{r}'_x$ **do**
    $\mathcal{R}_j^q \xleftarrow{add} r'_x, \quad \mu_{rq_j} \leftarrow \mu_{rq_j} \circ \mu(\overline{r}_x, \overline{r}'_x)$
  **end for**
**end for**
**for all** $n_x \in \mathcal{T}_i^q \cup \mathcal{L}_i^q$ **do** {*types and literals are kept unchanged*}
  $\mathcal{N}_j^q \xleftarrow{add} n_x$
**end for**
**return** $R_j$

# References


*[1]* http://www.w3.org/TR/rdf-sparql-query*.*

*[2] Christos Doulkeridis, Akrivi Vlachou, Yannis Kotidis, and Michalis Vazirgiannis. Peer-to-peer similarity search in metric spaces. In* VLDB*, 2007.*

*[3] Ronald Fagin, Amnon Lotem, and Moni Naor. Optimal aggregation algorithms for middleware.* J. Comput. Syst. Sci.*, 66(4):614–656, 2003.*

*[4] C. Gennaro, M. Mordacchini, S. Orlando, and F. Rabitti. MRoute: A Peer-to-Peer Routing Index for Similarity Search in Metric Spaces. In* Proceedings of the 5th International Workshop on Databases, Information Systems and Peer-to-Peer Computing (DBISP2P 2007)*, September 2007. To Appear.*

*[5] C. Gennaro, M. Mordacchini, S. Orlando, and F. Rabitti. Processing Complex Similarity Queries in Peer-to-Peer Networks. In* Proc. of SAC*, 2008.*

*[6] F. Mandreoli, R. Martoglia, W. Penzo, and S. Sassatelli. SRI: Exploiting Semantic Information for Effective Query Routing in a PDMS. In* Proc. of WIDM*, 2006.*

*[7] Federica Mandreoli, Riccardo Martoglia, Wilma Penzo, Simona Sassatelli, and Giorgio Villani. Sri@work: Efficient and effective routing strategies in a pdms. In* Web Information Systems Engineering - WISE 2007, 8th International Conference on Web Information Systems Engineering, Nancy, France, December 3-7, 2007, Proceedings*, pages 285–297, 2007.*

[8] *Federica Mandreoli, Riccardo Martoglia, Simona Sassatelli, and Wilma Penzo. Sri: exploiting semantic information for effective query routing in a pdms. In* Eigth ACM International Workshop on Web Information and Data Management (WIDM 2006), Arlington, Virginia, USA, November 10, 2006*, pages 19–26, 2006.*

[9] *Federica Mandreoli, Riccardo Martoglia, and Paolo Tiberio. Approximate query answering for a heterogeneous xml document base. In* Web Information Systems - WISE 2004, 5th International Conference on Web Information Systems Engineering, Brisbane, Australia, November 22-24, 2004, Proceedings*, pages 337–351, 2004.*

[10] *Moreno Marzolla.* `libcppsim`*: a simula-like, portable process-oriented simulation library in C++. In G. Horton, editor,* Proc. of ESM04, the 18th European Simulation Multiconference*, Magdeburg, DE, 2004. SCS–European Publishing House.*

[11] *Stefano Montanelli and Silvana Castano. Semantically routing queries in peer-based systems: the h-link approach.* Knowledge Eng. Review*, 23(1):51–72, 2008.*

[12] *Wilma Penzo. Rewriting rules to permeate complex similarity and fuzzy queries within a relational database system.* IEEE Trans. Knowl. Data Eng.*, 17(2):255–270, 2005.*

[13] *Jiri Pospichal. Fuzzy sets and fuzzy logic: Theory and applications. by george j. klir and bo yuan. prentice hall: Upper saddle river, nj, 1995, 574 pp, isbn 0-13-101171-5.* Journal of Chemical Information and Computer Sciences*, 36(3):619, 1996.*

[14] *Igor Tatarinov and Alon Y. Halevy. Efficient query reformulation in peer-data management systems. In* Proceedings of the ACM SIGMOD International Conference on Management of Data, Paris, France, June 13-18, 2004*, pages 539–550, 2004.*