future internet

Article

Reinforcement Learning Based Query Routing Approach for P2P Systems

Fawaz Alanazi [†] and Taoufik Yeferny [*,†]

College of Science, Northern Border University, Arar 73222, Saudi Arabia; falanazi@nbu.edu.sa

* Correspondence: yeferny.taoufik@gmail.com

† These authors contributed equally to this work.

Received: 19 November 2019; Accepted: 5 December 2019; Published: 9 December 2019

Abstract: Peer-to-peer (P2P) systems have offered users an efficient way to share various resources and access diverse services over the Internet. In unstructured P2P systems, resource storage and indexation are fully distributed among participating peers. Therefore, locating peers sharing pertinent resources for a specific user query is a challenging issue. In fact, effective query routing requires smart decisions to select a certain number of peers with respect to their relevance for the query instead of choosing them at random. In this respect, we introduce here a new query-oriented approach, called the reinforcement learning-based query routing approach (RLQR). The main goal of RLQR is to reach high retrieval effectiveness as well as a lower search cost by reducing the number of exchanged messages and contacted peers. To achieve this, the RLQR relies on information gathered from previously sent queries to identify relevant peers for forthcoming queries. Indeed, we formulate the query routing issue as the reinforcement learning problem and introduce a fully distributed approach for addressing it. In addition, RLQR addresses the well-known cold-start issue during the training stage, which allows it to improve its retrieval effectiveness and search cost continuously, and, therefore, goes quickly through the cold-start phase. Performed simulations demonstrate that RLQR outperforms pioneering query routing approaches in terms of retrieval effectiveness and communications cost.

Keywords: query routing; P2P; reinforcement learning; cold-start

1. Introduction

Peer-to-peer (P2P) systems have offered users an efficient way to share various resources and access diverse services over the Internet. They are more scalable, cost-effective, and more autonomous compared to distributed client/server systems [1]. In this respect, several P2P protocols and systems (e.g., Gnutella [2], Chord [3], Pastry [4] to cite a few), mainly used for file sharing, have been deployed over the Internet. The latter has attracted increasing interest among both researchers and computer networking professionals [5]. Existing systems are categorized with respect to their architecture as structured or unstructured systems. The former methods organize participating peers in a pre-defined structure, such as ring or tree, then use an appropriate hashing function to compute peer identifiers and place shared resources accordingly [4]. Locating peers holding pertinent resources for user search queries (i.e., search mechanism) in structured P2P systems is easy since the network overlay is structured and resources are distributed among peers according to known rules. However, in such a dynamic environment, peers join and leave the system more frequently, and therefore, maintaining the overlay structure is the main hindrance [6]. Instead of structuring the P2P overlay on a fixed topology as structured systems do, unstructured P2P systems build a free network structure, in which peers randomly organize themselves and locally manage their own content. Indeed, each peer joining the network establishes at random a set of links with its neighborhood. Hence, peers do not have information about shared documents of their neighbors or of other peers in the network [5,7–10]. Although unstructured systems efficiently manage the network dynamicity and fully support the autonomy of peers, the search mechanism in these latter still remains a challenging issue. Indeed, locating relevant peers sharing pertinent documents for a specific user query is of key interest. In fact, it requires smart query routing decisions.

Existing query routing approaches in unstructured P2P systems have relied on various mechanisms, such as pure flooding [2], controlled flooding [2], random walk [11,12], k-walker [12,13], and gossiping [5,14,15]. The latter is an interesting mechanism widely adopted by recent query routing approaches [5,9]. The main thrust of gossiping is to route the query to k neighbors chosen with respect to the likelihood that they hold pertinent documents for the query [5,9,15,16], instead of choosing them at random like controlled flooding, random walk, and k-walker do. Indeed, existing gossiping-based approaches defined specific scoring functions to select the k relevant neighbors with the highest score aka: "K-Neighbors-Selection (K-NS) problem." To this end, two broad categories of query routing methods have been proposed. They are known as content-oriented and query-oriented routing methods [9]. The former gather meta-data about neighbor collections to build a global index per peer, which is then exploited by a specific scoring function to rank neighbors with respect to their shared documents and the query content. Hence, the query will be routed to the first k neighbors with the highest scores. Query-oriented routing methods rely on data accumulated from previous sent queries to identify the k relevant expected neighbors for the forthcoming queries. To this end, different data mining and machine learning techniques have been used to define or learn the scoring function [9,16].

Query-oriented methods are more advantageous than content-oriented ones, since they only exploit available data of previously sent queries, therefore, they do not require extra network communications to build and maintain the routing indices (i.e., global index) [5,9,15,16]. In this respect, we introduce here a new query-oriented approach, called reinforcement learning-based query routing approach (RLQR). The main goal of RLQR is to reach high retrieval effectiveness as well as lower communications cost by reducing the number of contacted peers. The main contributions of this study are as follows:

1. We introduce a fully distributed query routing approach for pure unstructured P2P systems, in which each peer learns only from its local gathered data.
2. We introduce a new formulation for the K-Neighbors-Selection (K-NS) problem different from that of the existing approaches. Indeed, we consider that the natural model for this issue are formalizations from reinforcement learning, in which an agent learns by taking actions that produce rewards. The goal of the agent is to find a selection policy of actions that maximizes the cumulative reward. In the query routing setting, the forwarder peer could run a learning algorithm (e.g., agent) that selects k neighbors (e.g., actions) for each search query. Each selected neighbor yields a binomial reward which expresses its ability to produce pertinent documents to the query. Indeed, the agent must learn a K-NS policy to maximize the cumulative rewards leading to higher user satisfaction.
3. We address the cold-start issue during training, which is considered as the main hindrance of query routing approaches based on supervised machine learning or data-mining methods. Indeed, a cold start happens when a new peer joins the network. Existing query-oriented methods assume that each peer has submitted certain number of queries and received replies from relevant peers. Hence, the gathered data from previous queries are stored in the sender log file then used as training data. However, this assumption is not fulfilled in the case of a newly joined peer. Therefore, existing methods [7,9,15,16] required to randomly flood a certain number of queries to gather training data, which undoubtedly leads to achieving low performances during the training phase. To tackle this problem, we introduce here reinforcement learning into the neighbors selection algorithm. Indeed, our approach RLQR balances between exploration (pulling different neighbors to learn new information) and exploitation (i.e., pulling the neighbor with the highest

estimated reward based on previous queries). Doing so, RLQR improves the routing performance continuously and therefore goes quickly through the cold-start phase [7,17–19].

The rest of this paper is structured as follows. In Section 2, we review the related work. In Section 3, we first formalize the query routing issue as the reinforcement learning problem then we thoroughly describe our approach. The simulation settings and the evaluation of the proposed approach are presented in Section 4. Section 5 concludes this study and sketches of future directions.

2. Related Work

In the literature, several query routing methods for unstructured P2P systems have been introduced. Their ultimate goal is to locate all relevant peers sharing pertinent resources for user queries while reducing network traffic and the number of contacted peers. A first method was relied on pure flooding [2], in which the forwarder peer (i.e., a peer that initiates or receives a search query) must first search for pertinent documents that match the query keywords in its local shared collection of documents. If it holds pertinent documents for the query then it replies to the requesting peer by a query-hit message containing the list of retrieved relevant documents. The query-hit message is routed back to the requesting peer through the reverse path of the query. After that, the forwarder peer forwards the query to all neighboring peers that continue to forward it in the same way. The query propagation process is stopped by a peer whenever a Time-To-Live counter (TTL) reaches a certain specified value or it receives the same query from different neighbors.

Although the pure flooding method is straightforward and robust, it produces heavy network traffic and an excessive number of peers are contacted. Several methods, such as controlled flooding [2], random walk [11,12], k-walker [12,13], and gossiping [14,15] were proposed as an improvement of pure flooding. Indeed, these methods introduced different ways of selecting the neighboring peers to whom the query would be forwarded. In this respect, controlled flooding forwards the query to k neighbors selected at random instead of all neighbors like pure flooding does. Random walk could be seen as a special case of controlled flooding where the number of selected neighbors k equals 1. This method produces less network traffic than controlled flooding, however, it achieves low retrieval effectiveness. K-walker is a combination of controlled flooding and random walk, in which peer initiating the query forwards it to k neighbors (k walkers) selected at random, which continue to forward it to one random neighbor and so on until a relevant peer is located or the TTL value is reached. Gossiping is an interesting mechanism widely used in smart query routing approaches. The main thrust of gossiping is to route the query to k neighbors more likely having the requested resource. Indeed, these neighbors are selected with respect to the likelihood that they hold pertinent documents for the query [5,15,16], instead of choosing them at random like Random walk, K-walker and controlled flooding do. Indeed, existing gossiping-based approaches relied on specific scoring functions to select the k relevant neighbors with the highest score aka: “k-Neighbors-Selection (K-NS) problem”. To this end, two broad categories of query routing methods have been proposed. They are known as content-oriented [9] and query-oriented routing [9] methods. The former gather meta-data about neighbors’ collections, which are then exploited by a specific scoring function to rank neighbors with respect to their shared documents and the query content. Hence the query will be routed to the first k neighbors with the highest scores [20–22].

Query-oriented routing methods rely on accumulated data of previously sent queries to identify the k relevant expected neighbors for a given forthcoming query. To this end, various statistical, data mining or machine learning techniques have been used to define or learn the scoring function. Query-oriented routing methods are more interesting than content-oriented ones, since they only exploit available data about previously sent queries, therefore, they do not require extra network communications to build and maintain the routing indices (i.e., meta-data about neighbors’ collections). In this respect, in [23] authors suggested building a local peer ontology that stores knowledge extracted from previous queries. In the proposed scheme, Resource Description Framework (RDF) statements were used to represent the extracted knowledge. A statement may describe either data or conceptual information. In addition, each peer maintains meta-information about these statements (e.g., source of the statement, overall confidence in the statement and respective source peer, etc). The local ontology of each peer is then exploited by a specific k-neighbors-Selection (K-NS) algorithm to select the k relevant neighbors for the query. Kalogeraki et al. [13] introduced the Intelligent Search (IS) query-oriented method. IS maintains a routing table that stores neighbors profiles. They suggested a simple vectorial representation of each neighbor profile consisting of the most recent queries processed by the neighbor altogether with the number of query hits. Whenever a peer initiates or receives a query it performs an on-line ranking of its neighbors with respect to their profiles and the query content. The query is then routed to the first k relevant neighbors. Ciraci et al. [16] introduced a supervised machine learning algorithm, called Route Learning, to solve the K-NS problem. In the learning phase, the introduced routing algorithm accumulates data from sent queries and related query-hits. Indeed, a feature space for each neighbor is created. Furthermore, neighboring peers and forthcoming queries are respectively considered as classes and new objects to classify in a supervised multi-classification problem. Therefore, neighbors (i.e., classes) to which the query is assigned are considered as the k relevant neighbors. Arour et al. [9] relied on the Formal Concept Analysis (FCA) theory [24], which is a method of extracting interesting clusters from relational data, to implicitly extract user interests from previous sent queries and associated query-hits (e.g., query identifier, query terms, the downloaded documents and source peers). User interests are stored in a local knowledge base per peer. Indeed, user interests express strong relationships between previous queries, common terms, and related positive peers. To forward the query, a learning query routing algorithm first computes the similarity between user interests and the query keywords. Peers included in the closest user interests to query are then selected as the k relevant peers to forward the query to.

3. RLQR Approach

In this section, we introduce a new machine learning approach for query routing in unstructured P2P systems. We first formalize the query routing issue in unstructured P2P systems as a reinforcement learning problem then we thoroughly describe our approach.

3.1. Problem Formulation

In the literature, most of query routing approaches in unstructured P2P systems are based on gossiping mechanisms. As outlined in the related work, within gossiping the forwarder peer routes the search query to k relevant neighbors selected among a set of m neighbors (i.e., $k < m$) with respect to their relevance for the query. Indeed, existing approaches relied on specific different scoring functions to select the k elevant neighbors with the highest score. Our formulation for the K-Neighbors-Selection (K-NS) problem differs from the existing approaches. Indeed, we consider that the natural model for this issue are formalizations from reinforcement learning, in which an agent learns by taking actions that produce rewards. The goal of the agent is to find a selection policy of actions that maximizes the cumulative reward. In the query routing setting, the forwarder peer should run a learning algorithm (e.g., agent) that selects k neighbors (e.g., actions) for each search query. Each selected neighbor yields a binomial reward which expresses its ability to produce pertinent documents for the query. In fact, the agent must learn a K-NS policy to maximize the cumulative rewards leading to higher user satisfaction. At first glance it appears that modeling the K-NS issue as a Multi-Armed Bandit (MAB) problem, which is considered as one of the classical problems in RL, seems like a natural approach. However, most of the existing MAB algorithms are designed to select one among different actions (e.g., neighbors) while a K-NS algorithm must select, for each search query, k neighbors (e.g., actions). To address this issue, we introduce in the sequel a generic neighbor selection algorithm that adapts MAB algorithms to the K-NS problem. Figure 1 shows the global architecture of our approach, with P2P query routing terminology in black and corresponding RL terminology in red and italics.

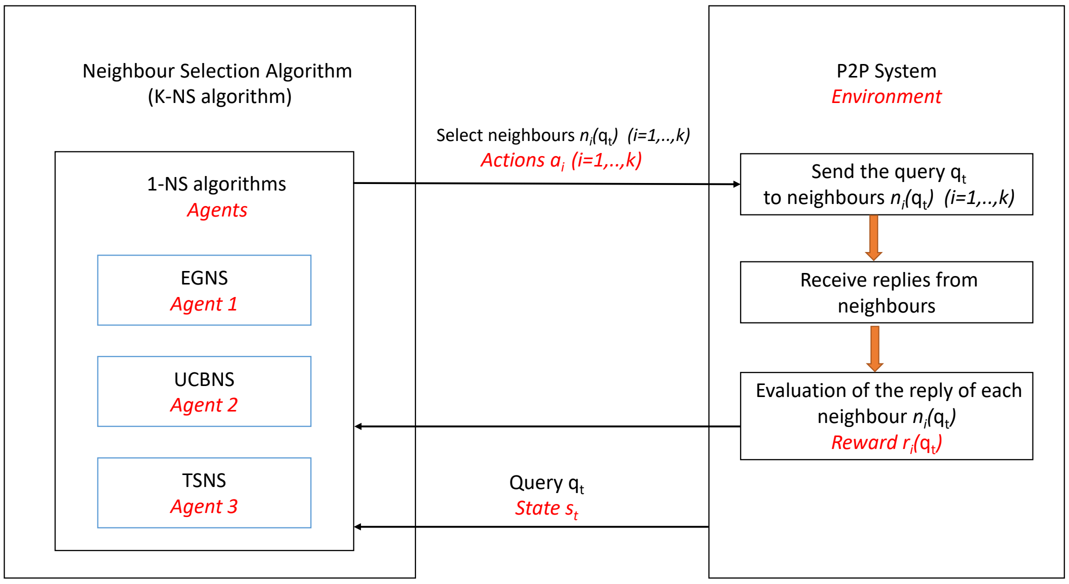
Neighbour Selection Algorithm
(K-NS algorithm)
1-NS algorithms
Agents
EGNS
Agent 1
UCBNS
Agent 2
TSNS
Agent 3
Select neighbours $n_i(q_t)$ $(i=1,..,k)$
Actions a_i $(i=1,..,k)$
Query q_t
State s_t
P2P System
Environment
Send the query q_t to neighbours $n_i(q_t)$ $(i=1,..,k)$
Receive replies from neighbours
Evaluation of the reply of each neighbour $n_i(q_t)$
Reward $r_i(q_t)$

Figure 1. The global architecture of the reinforcement learning-based query routing approach (RLQR) approach.

In the following, we first introduce three MAB-based learning algorithms for selecting one relevant neighbor for each search query, we will refer to these latter as 1-neighbor-Selection (1-NS) algorithms (Section 3.2), then we detail our generic K-NS algorithm (Section 3.3).

3.2. *1-Neighbor-Selection Algorithms*

Throughout this section, we assume that a forwarder peer has a set of $\mathcal{N}$ neighbors (e.g., actions) noted $n_1, \ldots, n_m$. For a given search query q_t, the forwarder runs a learning algorithm (e.g., an agent) to select one neighbor $n_i(q_t)$ then receives a reward $r_{n_i}(q_t) \in \{0,1\}$ from the selected neighbor $n_i(q_t)$ (i.e., $r_{n_i}(q_t) = 1$ if n_i returns pertinent documents to the query q_t, otherwise $r_{n_i}(q_t) = 0$). The goal of the agent is to learn a 1-neighbor-Selection (1-NS) strategy that maximizes the cumulative rewards, i.e., the sum of rewards over all processed queries during learning. We note that the agent doesn't have prior knowledge about the reward of each neighbor. Therefore, an efficient 1-NS strategy must balance between exploration (i.e., pulling different neighbors to learn new information) and exploitation (i.e., pulling the neighbor with the highest estimated reward based on previous queries). This will serve to enhance the routing performance continuously and, therefore, goes quickly through the cold-start phase [7,17–19].

In the following, we introduce an adaptation of three pioneering MAB algorithms for resolving the conflicting goals of exploration and exploitation.

3.2.1. EGNS: Epsilon-Greedy Based 1-NS Algorithm

The epsilon-Greedy is a MAB strategy that balances between exploiting the best action with a probability $(1-\varepsilon)$ and exploring randomly the available actions with a probability ε (ε is a specified threshold) [25]. We rely on this strategy to define our 1-NS algorithm EGNS. In fact, as illustrated in Algorithm 1, EGNS works by oscillating between (1) exploiting, with a probability $(1-\varepsilon)$, the neighbor with the highest estimated reward based on previous queries (see line 5 Algorithm 1) and (2) exploring at random, with a probability ε, among all the neighbors (see line 9 Algorithm 1). For example, if we set ε to 0.05, EGNS exploits 95% of the time the best neighbor with the highest average reward and selects at random any neighbor about 5% of the time. Worth noting that the average reward r_{n_i} of the selected neighbor $n_i(q_t)$ will be updated after receiving the reward $r_{n_i}(q_t)$ for the query q_t (see line 13 Algorithm 1).

Algorithm 1 EGNS: Epsilon-greedy-based 1-NS algorithm

```
Parameters:
    $q_t$: search query at trial $t$.
    $N = \{n_1, \dots, n_m\}$: set of all neighbors
    $R = (r_{n_1}, \dots, r_{n_m})$: average rewards of the $m$ neighbors
    $\varepsilon$: specified value of epsilon
    $n_i(q_t)$: the neighbor to be selected for $q_t$

 1: $value = random()$ # choose a random value
 2:
 3: if ($value > \varepsilon$) then
 4:
 5:     $n_i(q_t) = \text{argmax}_{n_i \in N}\{r_{n_i} \in R\}$
 6:
 7: else
 8:
 9:     $n_i(q_t) = random(n_i \in N)$
10:
11: end if
12:
13: update($r_{n_i}, r_{n_i}(q_t)$)
```

3.2.2. UCBNS: UBC-Based 1-NS Algorithm

Upper confidence bounds (UCB) is a family of MAB algorithms. Unlike the epsilon greedy algorithm, which pays attention only to the average reward of the actions, UCB algorithms rely on the average reward and on the confidence in the estimated reward's values of the selected actions [26]. It is worth mentioning that the UCB family of algorithms always chooses the best available actions and they do not randomly explore unlike epsilon greedy does. In this paper, we focus on the UCB1 algorithm and adapt it to define our 1-NS algorithm UCBNS. As illustrated in Algorithm 2, UCBNS starts by exploring all the m neighbors for the first m queries (see lines 1–9 Algorithm 2). Thereafter, it selects, for each query q_t, the neighbor $n_i(q_t)$ with the highest estimated reward and confidence value. This estimation is calculated with respect to the number of times the neighbor n_i was selected, denoted c_{n_i}, and its average reward r_{n_i} for previous queries (see line 11 Algorithm 2). Worth noting that r_{n_i} and c_{n_i} will be updated after receiving the reward of the selected neighbor $n_i(q_t)$ (see lines 13–15 Algorithm 2).

Algorithm 2 UCBNS: UBC-based 1-NS algorithm

```
Parameters:
    $q_t$: search query at trial $t$.
    $N = \{n_1, \dots, n_m\}$: set of neighbors
    $R = (r_{n_1}, \dots, r_{n_m})$: average rewards of the $K$ neighbors
    $C = (c_{n_1}, \dots, c_{n_m})$: vector storing the number of times we have selected each of the $K$ neighbors
    $n_i(q_t)$: the neighbor to be selected for $q_t$

 1: for all $n_i \in N$ do
 2:
 3:     if ($r_{n_i} == 0$) then
 4:
 5:         $n_i(q_t) = n_i$
 6:
 7:     end if
 8:
 9: end for
10:
11: $n_i(q_t) = \text{argmax}_{n_i \in N}\{r_{n_i} + \frac{log(\sum_{j=1}^{m} c_{n_j})}{c_{n_i}}\}$
12:
13: update($r_{n_i}, r_{n_i}(q_t)$)
14:
15: $c_{n_i} + +$
16:
```

3.2.3. TSNS: Thompson Sampling-Based 1-NS Algorithm

Thompson sampling (TS) [27] is an efficient algorithm used for addressing the exploration/exploitation trade-off in different research areas and especially for solving the MAB problem. In this respect, we extend the idea of Thompson sampling to define our TSNS algorithm for solving the 1-NS problem. Indeed, we assume that each neighbor $n_i \in \{n_1, \dots, n_m\}$ yields a success (i.e., a reward of 1) with probability $\theta_i \in [0, 1]$ and produces a failure (i.e., a reward of 0) with probability $1 - \theta_i$. The success probabilities $(\theta_1, \dots, \theta_m)$ of the m neighbors are unknown to the forwarder peer (i.e., the learner), but are fixed over time, and therefore can be learned from previous queries. We suppose that the learner starts with an independent prior belief over each θ_i and this latter is beta-distributed with parameters α_i and β_i, which denote the number of times n_i produces a reward of 1 (success) or a reward of 0 (failure), respectively. Therefore, the prior Probability Density Function (PDF) of the probability of success θ_i of each neighbor $n_i \in \{n_1, \dots, n_m\}$ is calculated as follows:

$$p(\theta_i) = \frac{\Gamma(\alpha_i + \beta_i)}{\Gamma(\alpha_i)\Gamma(\beta_i)} \theta_i^{\alpha_i - 1} (1 - \theta_i)^{\beta_i - 1}, \tag{1}$$

here Γ denotes the gamma function. It is worth mentioning that whenever a neighbor is selected and its reward is observed the distribution is updated according to Bayes' rule. The main idea of TSNS is to choose a neighbor according to its probability of being the best one. To do so, TSNS initially assumes that each neighbor n_i to have prior $Beta(\alpha, \beta)$, where $\alpha = \beta = 1$, on θ_i (see initialization part in Algorithm 3). It then samples a success probability estimate $\hat{\theta}_i$ of each neighbor n_i from its posterior distributions of θ_i, which is a beta distribution with parameters α_i and β_i (see lines 1–5 Algorithm 3). Thereafter, it selects, for the query q_t, the neighbor $n_i(q_t)$ with the largest estimate $\hat{\theta}_i$ (see line 7 Algorithm 3). Finally, after receiving the reward $r_i(q_t)$, TSNS updates the distribution parameters α_i and β_i of the selected neighbor $n_i(q_t)$ (see lines 9–11 Algorithm 3).

Algorithm 3 TSNS: Thompson sampling-based 1-NS algorithm

```
Parameters:
    $q_t$: search query at trial $t$.
    $N = \{n_1, \dots, n_m\}$: set of neighbors
    $\alpha = (\alpha_1, \dots, \alpha_m)$: alpha parameters of the $m$ neighbors
    $\beta = (\beta_1, \dots, \beta_m)$: beta parameters of the $m$ neighbors
    $n_i(q_t)$: the neighbor to be selected for $q_t$
Initialisation:
    $\alpha = (1, \dots, 1)$
    $\beta = (1, \dots, 1)$

 1: for all $n_i \in N$ do
 2:
 3:     sample $\hat{\theta}_i = Beta(\alpha_i, \beta_i)$
 4:
 5: end for
 6:
 7: $n_i(q_t) = \text{argmax}_{n_i \in N} \hat{\theta}_i$
 8:
 9: $\alpha_i = \alpha_i + r_{n_i}(q_t)$
10:
11: $\beta_i = \beta_i + r_{n_i}(q_t)$
12:
```

3.3. K-NS Algorithm

We introduce here a generic K-NS algorithm that relies on the proposed 1-NS algorithms (EGNS, UCBNS, and TSNS) to select the k relevant neighbors to forward the query to. The main idea is summarized in Algorithm 4. It consists simply of running one of the three introduced 1-NS algorithms (EGNS, UCBNS or TSNS) K times. As shown in Algorithm 4, the inputs of K-NS are the search query q_t at trial t, the set of all neighbors $N = \{n_1, \dots, n_m\}$ and the number of neighbors to be selected k. The output is a set S of relevant neighbors to forward the query q_t to. At each iteration $i = 1 \dots k$, K-NS relies on EGNS, UCBNS or TSNS to select from the set $N = \{n_1 \ldots n_m\}$ the neighbor n_j with the highest expected probability of success according to the used 1-NS algorithm (see line 3 Algorithm 4). The selected neighbor n_j at the iteration i, is then added to the set of relevant neighbors S and removed from the set of neighbors N (see lines 5–7 Algorithm 4).

Algorithm 4 K-NS algorithm

```
Input:
    $q_t$: search query at trial $t$.
    $N = \{n_1, \ldots, n_m\}$: set of all neighbors
    $k$: number of neighbors to be selected for $q_t$
Output:
    $S$: set of neighbors to be selected
Initialization:
    $S = \emptyset$:
1: for $i = 1 \ldots k$ do
2:
3:     select neighbor $n_j$ using a 1-NS learning algorithm (EGNS, UCBNS or TSNS)
4:
5:     $S = S \bigcup \{n_j\}$
6:
7:     $N = N \setminus \{n_j\}$
8:
9: end for
```

4. Performance Evaluation

In this section, we compare the retrieval effectiveness and the search cost of our routing approach RLQR vs Controlled Flooding (CF) [2] and Learning Peer Selection (LPS) [9]. It is important to note that we assess three RLQR instances, noted with respect to the used 1-NS learning algorithm: RLQR-EGNS, RLQR-UCBNS, and RLQR-TSNS. To this end, different simulation scenarios are performed using the PeerSim simulator [28]. The latter is an open-source Java simulator widely used for implementing and testing prototypes of P2P protocols. As the dataset, we use the "Big Dataset" collection [29]. The latter consists of 25,000 documents and 4000 queries distributed among 1000 peers using a benchmarking software developed by Zammali et al. [29]. The Time-To-Live of the query (i.e., the TTL parameter) and the number of neighbors to be selected by the forwarder peer (i.e., the k parameter) are set to 5 and 3, respectively.

4.1. Evaluation Measures

We use the **Recall** metric [30] in order to evaluate the retrieval effectiveness of RLQR vs its competitors. The recall $R(q_i)$ for the query q_i is defined as follows:

$$R(q_i) = \frac{RRD}{RLD}, \tag{2}$$

where, RRD and RLD are the numbers of relevant retrieved documents and the number of relevant ones, respectively. The cumulative average recall up to n sent queries (CAR_n) is defined as follows:

$$CAR_n = \frac{\sum_{i=1}^{n} R(q_i)}{n}. \tag{3}$$

To assess the search cost of the different routing algorithms we use the following metrics:

- **CP(q_i)**: it refers to the number of contacted peers for the query q_i. The Cumulative Average of the number of Contacted Peers up to n sent queries ($CACP_n$) is defined as follows:

$$CACP_n = \frac{\sum_{i=1}^{n} CP(q_i)}{n}. \tag{4}$$

- **Overhead(q_i)**: it refers to the number of exchanged messages for the query q_i. The Cumulative Average Overhead up to n sent queries (CAO_n) is defined as follows:

$$CAO_n = \frac{\sum_{i=1}^{n} Overhead(q_i)}{n}. \tag{5}$$

4.1.1. Retrieval Effectiveness of the Routing Algorithms

Figure 2 shows the evolution of the cumulative average recall of RLQR-EGNS, RLQR-UCBNS, RLQR-TSNS, CF, and LPS. As expected CF has low and stable recall (around 0.32) since queries are randomly flooded over the P2P network. We note that the recall of LPS is low (around 0.32) during the training phase then it improves after each update of the LPS' knowledge base. During the simulation, we periodically updated the knowledge base of each peer three times (i.e., after sending 1000, 2000 and 3000 queries). Moreover, we observe that the recall of our reinforcement learning algorithms RLQR-EGNS, RLQR-UCBNS and RLQR-TSNS increases rapidly, after sending a few queries, allowing them to outperform CF and LPS. Indeed, the cumulative average recall of CF, LPS, RLQR-EGNS, RLQR-UCBNS, and RLQR-TSNS for all sent queries is 0.17, 0.50, 0.64, 0.67, and 0.77, respectively. These encouraging results are owed to the fact that RLQR relies on reinforcement learning neighbor selection strategies (i.e., Epsilon Greedy, Upper Confidence Bounds, and Thompson Sampling) that efficiently balance between exploration and exploitation. Doing so, RLQR improves the routing performance continuously and, therefore, goes quickly through the cold-start phase. Moreover, we observe that Thompson Sampling-based neighbor selection algorithm RLQR-TSNS outperforms its competitors. This can be explained by the fact that RLQR-TSNS has a more sophisticated exploration strategy.

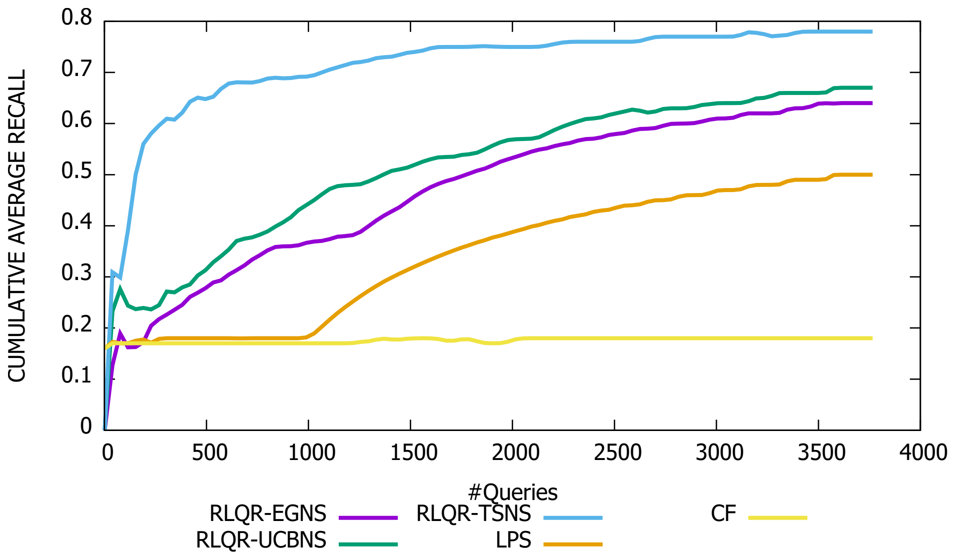

Figure 2. Evolution of the cumulative average recall.

4.1.2. Search Cost of the Routing Algorithms

Figures 3 and 4 illustrate the search cost of our query routing algorithms RLQR-EGNS, RLQR-UCBNS, and RLQR-TSNS compared with CF and LPS. Indeed, we observe that the cumulative average overhead and the number of contacted peers per query of CF are high and stable over the simulation time at about 324 and 275, respectively. Furthermore, we note that the cumulative average overhead and the number of contacted peers per query of LPS decrease after the training phase. Indeed, LPS starts with an empty knowledge and it relies on CF during the training phase (i.e., first 1000 queries), which leads to achieving lower performance as CF. However, the cost search of LPS improves after updating the knowledge base of each peer to achieve a cumulative average overhead and number of contacted peers per query around 300 and 255, respectively. Moreover, we observe that the search cost of our reinforcement learning algorithms RLQR-EGNS, RLQR-UCBNS, and RLQR-TSNS going down rapidly allowing them to outperform CF and LPS. Indeed the cumulative average overhead per query of RLQR-EGNS, RLQR-UCBNS, and RLQR-TSNS decreases from 324 to 287, 272 and 268, respectively. This is owed to the fact that our reinforcement learning routing algorithms locate more peers holding pertinent documents for the query. The latter stops the query flooding process to avoid cycles then returns positive replies to requesting peers.

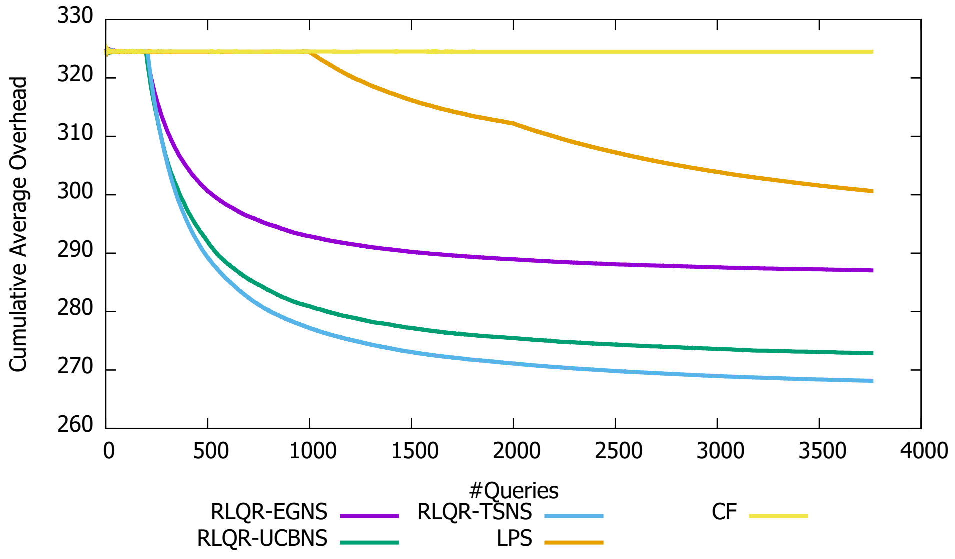

Figure 3. Evolution of the cumulative average overhead.

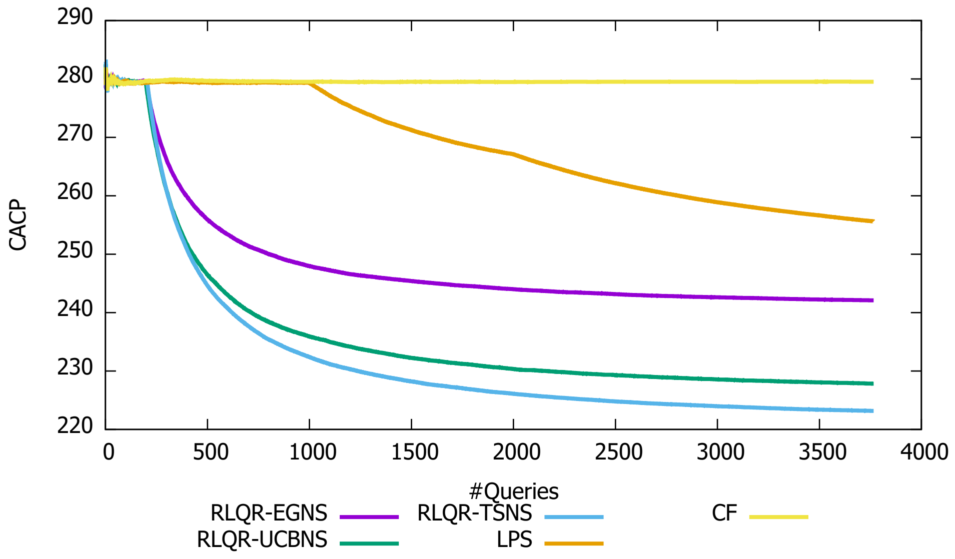

Figure 4. Evolution of the number of contacted peers.

5. Conclusions

In this paper, we have introduced RLQR a query routing approach for unstructured P2P systems based on the reinforcement learning theory. In this respect, we have formulated the query routing issue as the reinforcement learning problem and we have introduced a fully distributed approach for addressing it. In fact, we have introduced three reinforcement learning neighbor selection algorithms based on epsilon greedy, upper confidence bounds and Thompson sampling selection strategies. The introduced algorithms deal with the cold-start issue during the training stage, which allows them to improve the retrieval effectiveness and the search cost continuously and, therefore, goes quickly through the cold-start phase. Performed simulations have shown that our approach RLQR outperforms its competitors CF and LPS. Obvious pointers for future work are as follows:

- Studying the impact of the churn problem on the introduced neighbor selection strategies. Indeed, in P2P systems peers could join and leave the network at any time leading to frequent changes of neighbors links. This latter may have an impact on the performance of the neighbor selection algorithm.
- Introducing a reinforcement learning model that includes the context in the neighbor selection strategy to make the forwarding decision conditional on the state of the environment.

Author Contributions: Both authors (F.A. and T.Y.) contributed equally to the article.

Funding: This work was supported by the Deanship of Scientific Research at Northern Border University, Arar, Saudi Arabia, under Grant SCI-2018-3-9-F-8001.

Acknowledgments: This project was funded by Deanship of Scientific Research, Northern Border University for their financial support under grant no. SCI-2018-3-9-F-8001. The authors, therefore, acknowledge with thanks DSR technical and financial support.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Chernov, S.; Serdyukov, P.; Bender, M.; Michel, S.; Weikum, G.; Zimmer, C. Database selection and result merging in P2P web search. In Proceedings of the 3rd International Workshop on Databases, Information Systems, and Peer-to-Peer Computing (DBISP2P 2005), Trondheim, Norway, 28–29 August 2005; Lecture Notes in Computer Science; Springer: Heidelberg, Germany, 2005; Volume 4125.
2. Chawathe, Y.; Ratnasamy, S.; Breslau, L. Making gnutella-like P2P systems scalable. In Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, Karlsruhe, Germany, 25–29 August 2003; pp. 407–418.
3. Stoica, I.; Morris, R.; Liben-Nowell, D.; Karger, D.R.; Kaashoek, M.F.; Dabek, F.; Balakrishnan, H. Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications. *IEEE/ACM Trans. Netw.* **2003**, *11*, 17–32. [CrossRef]
4. Rowstron, A.; Druschel, P. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms, Heidelberg, Germany, 12–16 November 2001; Volume 2218, pp. 329–350.
5. Taoufik, Y.; Sofian, H.; Yahia, S.B. Query Learning-Based Scheme for Pertinent Resource Lookup in Mobile P2P Networks. *IEEE Access* **2019**, *7*, 49059–49068.
6. Deshpande, M.; Venkatasubramanian, N. The Different Dimensions of Dynamicity. In Proceedings of the 4th International Conference on Peer-to-Peer Computing (P2P'04), Zurich, Switzerland, 25–27August 2004; pp. 244–251.
7. Yeferny, T.; Arour, K. Efficient routing method in p2p systems based upon training knowledge. In Proceedings of the 26th International Conference on Advanced Information Networking and Applications Workshops, Fukuoka, Japan, 26–29 March 2012; pp. 300–305.
8. Yeferny, T.; Arour, K.; Bouzeghoub, A. An efficient peer-to-peer semantic overlay network for learning query routing. In Proceedings of the 27th IEEE International Conference on Advanced Information Networking and Applications (AINA), Barcelona, Spain, 25–28 March 2013; pp. 1025–1032.
9. Arour, K.; Yeferny, T. Learning model for efficient query routing in P2P information retrieval systems. *Peer-to-Peer Netw. Appl.* **2015**, *8*, 741–757. [CrossRef]
10. Yeferny, T.; Hamad, S.; Belhaj, S. CDP: A Content Discovery Protocol for Mobile P2P Systems. *Int. J. Comput. Sci. Netw. Secur.* **2018**, *18*, 28.

11. Lv, Q.; Cao, P.; Cohen, E.; Li, K.; Shenker, S. Search and replication in unstructured peer-to-peer networks. In Proceedings of the 16th International Conference on Supercomputing, New York, NY, USA, 22–26 June 2002; ACM: New York, NY, USA, 2002; pp. 84–95. [CrossRef]
12. Jia, Z.; You, J.; Rao, R.; Li, M. Random walk search in unstructured P2P. *J. Syst. Eng. Electron.* **2006**, *17*, 648–653.
13. Kalogeraki, V.; Gunopulos, D.; Zeinalipour-Yazti, D. A local search mechanism for peer-to-peer networks. In Proceedings of the Eleventh International Conference on Information and Knowledge Management, McLean, VA, USA, 4–9 November 2002; pp. 300–307.
14. Dietzfelbinger, M. Gossiping and broadcasting versus computing functions in networks. *Discret. Appl. Math.* **2004**, *137*, 127–153. [CrossRef]
15. da Hora, D.N.; Macedo, D.F.; Oliveira, L.B.; Siqueira, I.G.; Loureiro, A.A.F.; Nogueira, J.M.; Pujolle, G. Enhancing peer-to-peer content discovery techniques over mobile ad hoc networks. *Comput. Commun.* **2009**, *32*, 1445–1459. [CrossRef]
16. Ciraci, S.; Korpeoglu, I.; Ulusoy, Z. Reducing query overhead through route learning in unstructured peer-to-peer network. *J. Netw. Comput. Appl.* **2009**, *32*, 550–567. [CrossRef]
17. Li, L.; Chu, W.; Langford, J.; Schapire, R.E. A Contextual-bandit Approach to Personalized News Article Recommendation. In Proceedings of the 19th International Conference on World Wide Web WWW '10, Raleigh, NC, USA, 26–30 April 2010; pp. 661–670.
18. Wang, L.; Wang, C.; Wang, K.; He, X. BiUCB: A Contextual Bandit Algorithm for Cold-Start and Diversified Recommendation. In Proceedings of the 2017 IEEE International Conference on Big Knowledge (ICBK), Hefei, China, 9–10 August 2017; pp. 248–253.
19. Qiao, R.; Yan, S.; Shen, B. A Reinforcement Learning Solution to Cold-Start Problem in Software Crowdsourcing Recommendations. In Proceedings of the 2018 IEEE International Conference on Progress in Informatics and Computing (PIC), Suzhou, China, 14–16 December 2018; pp. 8–14.
20. Lu, J.; Callan, J. Content-based retrieval in hybrid peer-to-peer networks. In Proceedings of the Twelfth International Conference on Information and Knowledge Management, CIKM '03, New Orleans, LA, USA, 3–8 November 2003.
21. Kurid, H.A.; Alnusairi, T.S.; Almujahed, H.S. OBAME: Optimized Bio-inspired Algorithm to Maximize Search Efficiency in P2P Databases. *Procedia Comput. Sci.* **2013**, *21*, 60–67. [CrossRef]
22. Shen, W.W.; Su, S.; Shuang, K.; Yang, F.C. SKIP: An efficient search mechanism in unstructured P2P networks. *J. China Univ. Posts Telecommun.* **2011**, *17*, 64–71. [CrossRef]
23. Christoph, T.; Steffen, S.; Adrian, W. Semantic Query Routing in Peer-to-Peer Networks based on Social Metaphors. In Proceedings of the 13th International World Wide Web Conference, New York, NY, USA, 17–20 May 2004; pp. 55–68.
24. Ganter, B.; Wille, R. *Formal Concept Analysis: Mathematical Foundations*; Springer: New York, NY, USA, 1997.
25. Berry, D.A.; Fristedt, B. Bandit Problems. Sequential Allocation of Experiments. Monographs on Statistics and Applied Probability. *Biom. J.* **1987**, *29*, 20.
26. Agrawal, R. Sample Mean Based Index Policies with O(log n) Regret for the Multi-Armed Bandit Problem. *Adv. Appl. Probab.* **1995**, *27*, 1054–1078. [CrossRef]
27. Thompson, W.R. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika* **1933**, *25*, 285–294. [CrossRef]
28. Jelasity, M.; Montresor, A.; Jesi, G.P.; Voulgaris, S. The Peersim Simulator. March 2010. Available online: http://peersim.sf.net (accessed on 7 December 2019).
29. Zammali, S.; Arour, K. P2PIRB: Benchmarking framework for P2PIR. In Proceedings of the Third International Conference on Data Management in Grid and Peer-to-Peer Systems (Globe), Bilbao, Spain, 1–2 September 2010; pp. 100–111.
30. Makhoul, J.; Kubala, F.; Schwartz, R.; Weischedel, R. Performance Measures For Information Extraction. In Proceedings of the DARPA Broadcast News Workshop, Herndon, VA, USA, 28 February–3 March 1999; pp. 249–252.