An Effective Path-aware Approach for Keyword Search over Data Graphs

***Asieh Ghanbarpour*[1], *Hassan Naderi*[2], *Soheil Zaremotlagh*[3]**
1- Computer Engineering Department, University of Sistan and Baluchestan, Zahedan, IRAN. (ghanbarpour@ece.usb.ac.ir)
2- Computer Engineering Department, Iran University of Science and Technology, Theran, IRAN.
3- University of Sistan and Baluchestan, Zahedan, IRAN.

Received (2020-03-13) Accepted (2020-08-05)

Abstract: *Keyword Search is known as a user-friendly alternative for structured languages to retrieve information from graph-structured data. Efficient retrieving of relevant answers to a keyword query and effective ranking of these answers according to their relevance are two main challenges in the keyword search over graph-structured data. In this paper, a novel scoring function is proposed, which utilizes both the textual and structural features of answers in order to produce a more accurate order of answers. In addition, a query processing algorithm is developed based on information spreading technique to enumerate answers in approximate order. This algorithm is further improved by allowing a skewed development toward more promising paths and enables a more efficient processing of keyword queries. Performance evaluation through extensive experiments on a standard benchmark of three real-world datasets shows the effectiveness and efficiency of the proposed algorithms.*

Keywords: *Information retrieval, Database, Keyword search, Relevant answers, Information spreading.*

How to cite this article:
Asieh Ghanbarpour, Hassan Naderi, Soheil Zaremotlagh. An Effective Path-aware Approach for Keyword Search over Data Graphs. J. ADV COMP ENG TECHNOL, 6(3) Summer 2020 : 169-186

I. INTRODUCTION

In recent years, keyword search has attracted the attention of researchers as an effective method to retrieve information from different datasets which could be modeled as a graph. The success of keyword search is rooted in its simplicity in expressing a query (a set of keywords). It enables the user to search for his/her needed information without knowing a specialized query language (e.g. SQL, SPARQL or XQuery) or having a prior knowledge about the underlying data structure (e.g. tables, docs or properties) [1]. An answer to a keyword query is a set of nodes connected as a sub-tree or a sub-graph covering all the keywords of the query. It represents how the nodes containing query keywords are interconnected in the dataset [2]. For example, consider Fig 1. This figure shows a part of a geographical database modeled as a graph. Any node in this graph represents a tuple in the dataset, and any edge shows a connection made between the connected tuples by foreign keys. Suppose a user queries the dataset by Q={Turkmenistan,Uzbek} to know the relationship between the two queried keywords. These keywords are not covered together in any of the graph nodes. Instead, two minimal collections of nodes cover the wanted keywords as they are shown in Fig 1. Collection A_1 is an answer to query Q and means “Uzbek is the name of an Ethnic group in Turkmenistan including 9% of the total population”. Collection A_2 is another answer to the query and means “Turkmenistan is bordered by a country named Uzbekistan in which Uzbek is the official language”.

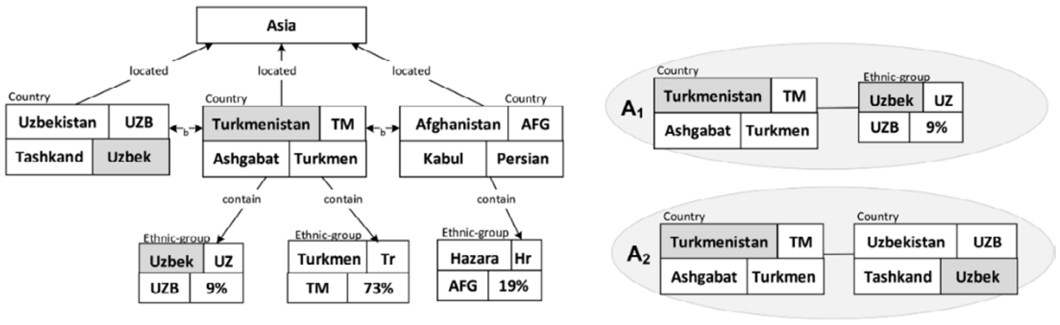

Fig1. A modified part of Mondial dataset and two answers in response to query Q={Turkmenistan,Uzbek}.

A keyword search system consists of two main subsystems: a search subsystem to retrieve relevant answers to given query, and a ranking subsystem to rank the retrieved answers.

The search subsystem has an effective impact on the efficiency of the system. According to a comprehensive study, the search methods which were proposed on the schema-free graphs can be divided into four categories: cluster-based methods, subgraph-based methods, Lawler-based methods and virtual document-based methods. In the cluster-based methods, the keyword nodes are grouped into some initial clusters according to the keywords they covered. These clusters are expanded based on different strategies to reach common nodes. Detected answers by this way are in the form of star-trees in which leaves contain keyword nodes of different types. BANK [13], bidirectional [14] and BLINKS [5] are samples of such methods. Although these systems are fast in detecting answers, they are unable to retrieve non-star answers. Therefore, they present an incomplete list of answers to the user.

In the subgraph-based method, the search is initiated by the keyword nodes. In this method, the low-weights subgraphs are incrementally merged until reach answers. DBPF [4] is a dynamic programming implementation of this method. In DBPF, each edge is just observed in one answer. Therefore, a large number of answers are lost.

The Lawler-based methods produce a complete set of answers [6, 7]. In these methods, all the search space is first searched (by a system such as DBPF) to find the lowest weight answer. The search space is then divided to subspaces and each subspace is searched separately to find the next answer. Since the generated subspaces may differ only in one edge with each other and with the base graph, the search on them is done with the same upper bound complexity as the search on the overall graph. These expansive repetitive searches make Lawler-based methods empirically inapplicable on the large graphs.

In the virtual document-based methods more focus is on the text processing of subgraphs. In these methods, the nodes of graph are mapped to virtual documents and then processed with existing text processing methods. Graph-LM [16] and KESOSASD [17] are samples of such methods. Relying on the text retrieval processes which have been evolved a lot over the years [18], the virtual document-based methods could retrieve answers very quickly and accurately. However, these methods need an extra memory to save virtual documents. In addition, the radius of answers generated by these methods is limited to the size of considered virtual documents.

In response to a query, numerous answers may be retrieved by the search subsystem. Therefore, a ranking step is necessary to provide more relevant answers to the user.

A ranking function is usually used to evaluate the relevance of answers to a given query [3]. Some of the proposed ranking functions in the literature use only the structural attributes of answers to rank them [4-7]. It is obvious that these functions are unable to differentiate among the answers with the same structure but different content. A group of methods tried to solve this problem by considering the IR-style features (commonly TFIDF) in addition to the structural features of answers in the ranking efforts [8-12].

Although these methods are usually more successful than the first group, they are still unable to detect some differences between answers. For example, consider two answers shown in Fig 1. These answers are structurally similar. On the other hand, the TFIDF of the two keywords in both of the answers are similar. However, A_1 covers the queried keywords in more important attributes than A_2, and it should be considered more relevant to the user's query. This priority cannot be detected by the second group of methods because realizing it needs a semantic analysis on the text information of nodes up to the attribute level. One of our purpose in this paper is to present a scoring function which is able to differentiate answers with a high degree of distinction.

In this paper, two greedy-based keyword systems are proposed, which use a new scoring function during the search process to retrieve top-k answers. The proposed systems provide answers in an approximate order of their final ordering. This ability allows the systems to present answers immediately after they are retrieved. As a result, it reduces the response time of the systems significantly in comparison to those of the other systems.

The main contributions of this work are summarized as follows:

- To provide a more accurate order of relevant answers, we propose a new scoring function to observe both the content-based and the structure-based relevance of answers to the query. The content-based relevance of answers is defined based on a new notation of term weights taking the attribute-level information into consideration. Using this level of textual information beside using the structural information in ranking of answers makes the proposed function more powerful to distinct answers.
- For effective finding of top-k answers, we propose Blind Keyword Search (BKS) system which retrieve answers in an approximate order of their final ranking. This algorithm provides a solution based on information spreading technique [19, 20] to retrieve top-k answers. In addition, special nodes (as a new notation) are used for an initial pruning on the search space. A bounded approximation ratio is proved for the result list of this system.
- To improve the efficiency of BKS, we then propose an enhanced search system named Informed Keyword Search (IKS). The algorithm estimates the probability of answer existence in all paths in order to follow the more probable paths earlier than the others. It can retrieve the answers more efficiently than BKS but has a slight lowering in effectiveness.

In the experiments, we show the effectiveness of the approximate ranked lists of answers presented by the proposed systems in comparison to the ranked list of the other systems. In addition, we show the efficiency of IKS in retrievement of top-k answers in terms of response time and execution time. The rest of the paper is structured as follows: In Section 2, related works are discussed. Then, in Section 3, formal problem statements are defined and the proposed scoring function are provided. Two query processing algorithms for solving keyword search problem are introduced in Section 4. The evaluation framework and the results of the experiments are explained in Section 5. Finally, in Section 6 the paper is concluded.

II. RELATED WORKS

Conducted research on keyword search were followed in four categories: Keyword search on relational databases [2, 4, 8, 9, 21, 22], Keyword search on XML databases[23, 24], Keyword search on Semantic Web [25-27] and Keyword search on schema-free graphs [5, 6, 11, 14, 15, 28-30]. All the relational, XML and RDF datasets are associated with predefined schemas. Using the schema facilitates the determination of the query meaning and helps the effective search of data. Keyword search over schema free graphs is facing more challenges because of having less prior knowledge about the examined data. In the presented paper, we focused on the schema-free graphs because there are many graphs for which the scheme is not defined. In the following, some of the works that are more similar to our work are described. In all of these works, the search is started from keyword nodes and followed using a traversing strategy to reach answers.

BANKS [13], Bidirectional [14] and BLINKS [5] are cluster-based methods which follow forward expansion, bidirectional expansion and cost-based expansion respectively to find answers. BLINKS [5] partitions graph data into some blocks and utilizes a bi-level indexing to process keyword queries. One group of indexes is used to travel between the blocks and another is used to access the data within the blocks. In retrieving an answer, BLINKs begins from a keyword node and searches its block with the help of intra-block indexes. If searching for an answer is expanded to the neighborhood blocks, it utilizes multiple cursors and sends each of them to a neighbor block to continue the search in them. According to the claim of the authors, BLINKS is m-optimal, where m is the number of input keywords. BANKS, Bidirectional and BLINKS just retrieve star-trees as answers and ignore other ones. In addition, these systems could not provide any answer until retrieving all the relevant ones. Ease which is introduced in [31] defined the concept of r-radius sub-graphs on undirected graph and retrieved them as answers of a keyword query. It used two indexes for storing data, the first index saved the structural distance between each pair of keywords and the second index stored the contained r-radius sub-graphs for each keyword. This approach contained a ranking step to sort retrieved answers according to their structural compression and some IR measures. In this method, the size of final answers is limited to the size of initial processed sub-graphs. So, there may be answers that are never retrieved. Virgilio in [32] proposed an approach based on the paths which led to the keyword nodes. In this approach, a path of graph was selected in each step to be added to an incomplete answer. The paths were grouped based on their template and assigned by a score. The score of paths was not only used in the expanding phase, but also in the final ranking of answers. In [6], [7], a Lawler-based method were employed to retrieve answers. Kargar and An in [7] focused on finding answers in the form of r-cliques. An r-clique is a set of nodes which covers all the query keywords and whose shortest path between each two nodes is not greater than r. r is a user-defined parameter and indicates the maximum value of the answer's diameter. Restricting answers to r-clique ones is main disadvantage of this system. The authors claimed that in the worst case, the weight of an answer produced by their algorithm is twice the weight of the optimal answer. In [6], the authors focused on retrieving top-k non-duplicate answers. The duplicate answers are ones covering the same set of keyword nodes. Such answers are retrieved by dividing the search space into some disjoint subspaces and performing an iterative search on the subspaces. The best answer in each subspace is obtained and used to produce the best global answer. The subspace which produces the best global answer is further divided into sub-subspaces and the best answer among its sub-subspaces is used to compete with the best answers in other sub-spaces in the previous level in order to find the next best global answer. Although Lawler-based methods are highly effective, the need for expansive repetitive searches makes them practically inapplicable on large graphs.

Nguyen and Cao [33] presented an approach that selected the top-k data sources from potentially numerous data sources. Their method derives information patterns from each data source as succinct synopses that act as representatives of the corresponding data sources. The patterns (sub-graphs) are then scored based on their relevance to the given query using a structural-aware ranking function. As mentioned earlier, ranking answers based solely on their structural features is one of the main disadvantages of such methods. The papers [16, 34] proposed a virtual document-based method to retrieve answers. In this method, any subgraph is mapped to a virtual document (VD) by concatenating the textual content of its nodes. Therefore, the problem of searching on the graph data is reduced to the problem of searching on a set of documents. These papers use language models to score retrieved answers. Although the search on documents could perform effectively, the size of answers is restricted to the size of virtual documents. In addition, an extra memory is needed to build a layer of documents. A new type of keyword search query, ROU-query, was defined in [35]. It utilized input keywords in three categories: required, optional, and unwanted, and returned nodes of the underlying graph whose neighborhood satisfied the keyword requirements. It applied a new data structure named query induced partite graph (QuIP) to capture the constraints related to the neighborhood size and unwanted keywords. Due to the limitation of the result list to a specific form of answers, this system has achieved good efficiency but its effectiveness has decreased. The authors proposed a family of algorithms which took advantage of the information in QuIP for efficient evaluation of ROU-queries. Park and Lim in [11] proposed an approach to aggregate best keyword nodes gained from a pre-computed process in order to produce top-k relevant answers in an approximate order. They used a queuing system over the data extracted from an extended inverted index. The employed inverted index stores comprehensive neighborhood information between nodes. Using this index, the answers of a keyword query are retrieved by sharing the neighborhood information of nodes. The examined method results in low execution time but high space complexity because of the high volume of indexing. On the other hand, it prefers more extended and relevant answers having more coverage of keywords instead of minimal answers.

In the presented paper, a keyword search method is introduced by considering both the time and space complexities, which retrieves minimal answers without any restrictions on the shape of the answers.

III. PROBLEM STATEMENT AND PRELIMINARIES

Suppose an un-directed un-weighed graph G=(N,E) where N is a set of nodes and E is a set of edges. Each node $n_j \in N$ includes a list of attribute-value pairs as $n_j=\{(A_1,v_{j1}),\ldots,(A_n,v_{jn})\}, n\geq 1$, where A_i shows an attribute and v_{ji} shows the value of attribute A_i in node n_j. The edges represent the semantic relationships between pairs of nodes. Such graphs are used to model different types of unstructured, semi-structured, and structured datasets.

The problem of keyword search on the graph is stated as follows. Given a graph-structured database G and a query $Q=\{q1,q_2,\ldots,q_{|Q|}\}$, retrieve top-k answers from G in response to the query in a ranked order.

Definition 1 (Answer) Consider a keyword query Q, over graph *G*. An answer in response to Q is a minimal subgraph a_g from *G* such that for every keyword of the query there is at least one node in a_g covering the keyword. Minimality requires an answer not to have a proper sub-graph that also covers all the query keywords.

The answers are ranked based on their relevance score to the query, and the top-k ranked answers are presented to the user. Even for a simple definition of the scoring function f, an efficient enumeration of answers might not be possible. A formal notion of enumeration in an approximate order is defined as follows [36].

Definition 2 (θ-approximate order) Let answer a_g with less $f(a_g)$ is considered as the better answer. Enumerating answers in an θ-approximate order means that if answer a_g precedes answer a_g', then a_g is worse than a_g' by at most a factor of θ. More formally, the sequence of all answers $a_1,a_2,\ldots,a_n$ will be in θ-approximate order if $f(a_i)\leq\theta.f(a_j)$ for all $1\leq i\leq j\leq n$.

An algorithm that enumerates the answers in a θ-approximate order can find a θ-approximation of the top-k answers by terminating the process after outputting the k-th answer.

1. Scoring function

The search results of a keyword query are a collection of graphs. To rank these answers, the graph topology beside the textual content of nodes should be considered. The topology of a graph shows how the container nodes of keywords are close together. One of the most common ways of answer scoring in keyword search works is using the Steiner-based weights [4, 22, 37]. However, these weights are not suitable for measuring the closeness of answer nodes when the answer is in the form of a subgraph. That's because, in a subgraph, adding one edge leads to increasing the relationships between nodes and will bring them closer. It is while adding an edge increases the Steiner-based weight and subsequently causes the resulted answer is ranked at a lower level. In the presented work, we measure the closeness of nodes through two factors: the number of nodes and the diameter of the answer subgraph. The diameter of a subgraph is the greatest distance between any pair of nodes. Accordingly, the structural relevance of answer a_g is defined as follows,

$$SR_{sg}(\mathrm{a_g}) = \left(|a_g| \times \alpha^{d(a_g)}\right)^{-1} \tag{1}$$

Where $|a_g|$ shows the number of nodes of a_g and $d(a_g)$ denoted the diameter of a_g. Besides, α is a parameter to tune the impact of the two factors.

On the other hand, we define a content-based relevance score for any answer to be used together with a structural relevance score to improve the accuracy of the final ranking of answers. The keywords in a node are organized under a set of attributes with various levels of importance. The relevance of a node to a keyword depends on both the importance of the node attributes to which the keyword belongs and the importance of the keyword in those attributes value. The importance of a keyword w in a text value v has been studied extensively in the IR methods. In the proposed scoring function, the pivoted normalization weighting [8] is used which has been defined as follows,

$$CR_A(w,v) = \frac{1+\ln(1+ln(tf_v(w)))}{(1-s)+s\times\frac{dl_v}{avgdl}} \times \ln\left(\frac{N_v}{df_v(w)+1}\right) \tag{2}$$

here $tf_v(w)$ shows raw frequency of keyword w in the text v, dl_v denotes the length of v, avgdl represents the average length of attribute values, N_v denotes the total number of text values and $df_v(w)$ shows the number of text values in which w exists. Parameter s is a constant and is usually set to 0.2.

The importance of attributes is measured by assigning a weight to any attribute. These weights are context-dependent ones which could be determined by an expertise analysis or by a domain expert. Considering the weights of attributes, the relevance of node n_j to keyword w is defined as follows:

$$\mathrm{CR_N}(\mathrm{w},\mathrm{n_j}) = \sum_{A_t\in\mathbb{A}} AttributiveWeigth(A_t) \times CR_A(\mathrm{w}, v_{jt}) \tag{3}$$

where $\mathbb{A}$ shows the set of attributes known in

the corpus and v_{jt} denotes the value belonging to the domain of A_t in node n_j. It should be noted that $\sum_{A_t\in\mathbb{A}} AttributiveWeigth(A_t) = 1$.

According to the score of nodes, the content-based relevance score of answer a_g concerning query Q is defined as follows:

$$CR_{sg}(a_g|Q) = \sum_{q_i\in a_g, q_i\in Q} \max_{n_j\in a_g} CR_N(q_i,n_j) \tag{4}$$

The overall relevance score of answer a_g with regard to query Q is defined as a combination of its textual and structural relevance scores as follows:

$$Score(a_g|Q) = SR_{sg}(a_g) \times CR_{sg}(a_g|Q) \tag{5}$$

Where $SRsg(a_g|Q)$ and $CR_{sg}(ag|Q)$ shows the structural and textual relevance score of the answer concerning query Q, respectively.

IV. PROPOSED SEARCH ALGORITHMS

In this section, two search algorithms are proposed to retrieve top-k answers of a keyword query. These algorithms employ a smoothed form of the scoring function (proposed in section 3.1) to present answers in an approximate order of their final ranked order.

1. Blind Keyword Search (BKS)

Blind Keyword Search is a query processing model developed based on the information spreading technique to retrieve relevant answers to a given query. In this model, the spread of information is initialized from some query-dependent seed nodes. The information is diffused in the graph in a breadth-first stepwise manner until some nodes receive all the information needed to cover the query. The information of each node participating in the search process is packed to a holding list (HList) containing a

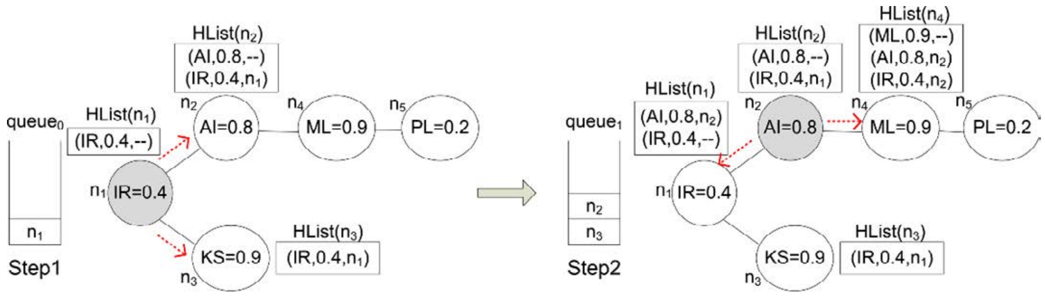

Fig 2. Two steps of processing Q={IR,AI,ML,PL} by BKS (n_1 is a seed node).

union of the queried keywords saved in the node or received from the other nodes. A keyword in a holding list is stored along with its weight in the original node and the ID of the node from which it was received.

In any step i^{th} ($1 \leq i \leq r$) of the search process, a priority queue ($queue_i$) is considered that stores the list of nodes that should be activated in step i^{th}. When a node is activated, the information of its holding list (mines the information which is received in step i^{th}) is spread to adjacent nodes.

The holding list of any adjacent node is updated based on the received information so that if it includes a keyword and again gets it from an active node, the alternative weight and ID of the active node are saved for the existing keyword. Otherwise, the received keyword along with its associated weight and the active node ID is saved as a new entry in the HList of the hosting node. An active node would be deactivated after spreading information although it could be activated again later. The union of receiver nodes in step i^{th}comprises the entries of the next step priority queue ($queue_{i+1}$). A transition between the steps occurs whenever all the nodes in the lower step have spread their information. The search process continues until k answers are retrieved or a predefined number of steps is reached. This number which is named as retrieval depth(r) is set to the maximum radius of the expected answers. Two steps of query processing by BKS are shown in Fig 2. The activated node is highlighted in this figure.

An answer is detected by a query processor if all the queried keywords are observed in the holding list of a node. This node is tagged as a detector node. The answer related to a detector node is retrieved by starting at the detector node and following the node IDs saved in the holding list of consecutive nodes in a backward manner. After retrieving an answer, the holding list of its detector node is re-updated and all information received from the other answer nodes is tagged as archive information. Archive information of a holding list will never be spread to the other nodes. This prevents to re-generate the retrieved answers in the other nodes.

Now suppose a node that has been previously tagged as a detector node receives new information. In this case, any combination of new information with the archived ones forms a new answer. Such answers are detected and added to the result set. The new information received in the detector node is then added to the archive part of the holding list.

Using sequential priority queues in BKS algorithm helps to maintain an activation order among nodes according to their distance to the seed nodes. It is a way to observe an approximate retrieval order of answers by increasing their radius. This order could be improved by taking the content-based relevance score of answers into account. The holding list of a node represents summarized information of the subgraph leading to the node. Therefore, to maintain a content-based order among the traversed subgraphs, a score named prestige is defined for each node, and the nodes of each priority queue are sorted based on decreasing order of their prestige. Prestige is a query-dependent score which is computed for every node n_d considering query Q as follows:

$$Prestige(n_d|Q) = \frac{1}{|Q|+1}\Bigg[|HList(n_d|Q)| + \prod_{q_i \in HList(n_d|Q), q_i \in Q} \max_{HList(n_d, q_i|Q)} CR_N(q_i, n_d)\Bigg] \quad (6)$$

where $HList(n_d|Q)$ denotes the set of unique keywords of Q which are covered by the holding list of n_d and $HList(n_d,q_i|Q)$ shows the set of entries for keyword $q_i \in Q$ in the holding list of n_d. Using the max function causes any keyword to contribute only once and with its best score in the calculation.

Suppose β denotes the average branching factor of nodes and $\tau(q_s)$ shows the number of seed nodes. To retrieve all answers with a radius equal to or lower than r, the BKS algorithm has a time complexity of $O(\tau(q_s).\beta^{r+1})$.

Theorem 1: BKS enumerates answers in the exact order of their radiuses and in $\frac{2r}{2r-1}$-approximate order of their diameters.

Proof: an answer is detected when the information of the farthest keyword node reaches the center node of the answer. The information of the farthest keyword node to the center node of an answer with radius r reaches the center node in step r^{th}. It implies that any answer with radius r is detected in the corresponding step of spreading. Since in BKS, spreading is performed in a regular stepwise manner, the answers would be generated in the exact order of their radiuses.

Among the answers with the same radius, the minimum and maximum diameters are ($2r$-1) and $2r$, respectively. Therefore, in the situation where the arrangement of answers is discussed based on their diameters, observing a displacement of $\frac{2r}{2r-1}$ among the answers with the same radius would be expected. Since among the answers of different groups, the arrangement of answers based on their diameter is true, it could be said that BKS presents answers in $\frac{2r}{2r-1}$-approximate order of their diameters.

Special nodes

In many keyword search algorithms, keyword nodes are considered as starter nodes and expanded until reach answers. The runtime of this strategy is from $O(\tau(q_s).\beta^{r+1})$ to retrieve all the answers with a radius equal to or lower than r over graph G with average branching factor β. However, many of these nodes may never lead to an answer within a limited range of expansion. As an example, suppose query $Q=\{q_1,q_2,q_3\}$. Let q_1 and q_2 be frequent keywords that have occurred in more than a thousand nodes, while q_3 is a special keyword in a professional background that has just appeared in some nodes. Commonly, the search begins from all the keyword nodes (more than two thousand) and continues by expanding all of them until detecting top-k answers. However, each comprehensive answer should cover at least one node containing q_3. Therefore, it would be faster if the search is done only in the neighborhood of the nodes containing q_3.

Definition 4 (Special node) Suppose query Q and graph G. A node is named special if it covers a keyword of query Q and if the frequency of that keyword in the nodes of G is not larger than those of other queried keywords in the same graph.

The second strategy is to consider special nodes as seed nodes. In this strategy, the search will proceed unilaterally and the same set of answers will be identified after $2r$ steps. In this case, the runtime complexity of BKS will be from $O(\tau(q_{sp}).\beta^{2r+1})$ where $\tau(q_{sp})$ shows the number of special nodes. In the proposed algorithms, an optimal strategy is followed to set the seed nodes. For every query, if the frequency of one queried keyword is much lower than those of the other queried keywords such that $\tau(q_{sp}).\beta^{2r+1} < \bigcup_{i=1,i\neq sp}^{|Q|} \tau(q_i).\beta^{r+1}$,the special nodes will be selected as the starter nodes for processing the query. Otherwise, the keyword nodes will be employed as starter nodes. Using this strategy causes an early pruning of the search space.

2. The Modified Search Algorithm (Informed Keyword Search)

The baseline model described in Section 4.1 performs a comprehensive search at the neighborhood of seed nodes. This model is not efficient since the number of contributing nodes in the search process increases exponentially by incrementing the depth of search. Informed keyword search (IKS), concerning the efficiency of the baseline method, is a more intelligent version of BKS which follows a purposive search to the paths having more potential to reach an answer. Fig 3 shows a schematic overview of BKS and IKS processes in detecting the first answer. The nodes of this answer are shown by filled circles and the nodes which have been activated at least once are shown by bold circles. The breadth-first manner of BKS in the traversing graph is evident in Fig 3 while IKS searches graph in a best-first manner.

In IKS, the query processor heuristically follows the paths with more signs of the existence of answers instead of following all the paths until a specified level. Since the information of a path is saved in the HList of the path nodes, a calculation on the content and position of nodes could be properly used in determining the priority of paths. Therefore, the problem of specifying a priority order among some paths is reduced to the problem of determining an activation order among the path's nodes. In IKS model, there is only one priority queue to determine the order of nodes to activate. This queue is initialized with a set of seed nodes that are sorted in decreasing order of their prestige.

The unique priority queue provides an overall view over all candidate nodes to activate and makes it possible to select every node in each level to be activated. The prestige score is the only affecting factor in determining the activation order of nodes. So, it should properly reflect the content and position of nodes. The existence of more query keywords during a path increases the possibility of reaching an answer. Therefore, the prestige of a node is defined in direct proportion to the number of keywords covered by the HList of the node. However, to observe an approximation order on the final list of answers, it is essential that the more compact answers with shorter paths are detected earlier. This implies that the progress toward special paths (although promising) is controlled by an upper bound. The control on the length of traversed paths is achieved by contributing the level of nodes in the prestige calculation. By assuming that the initial prestige of any seed node is calculated by (6), the prestige of any receiver node n_d is calculated as follows:

$$Prestige(n_d|Q) = \max\left(Prestige(n_d|Q), \frac{|HList(n_d|Q)| + \prod_{q_i \in HList(n_d|Q), q_i \in Q} \max_{HList(n_d, q_i|Q)} CR_N(q_i, n_d)}{(|Q|+1) \times \alpha^{Level(n_d)}}\right) \tag{7}$$

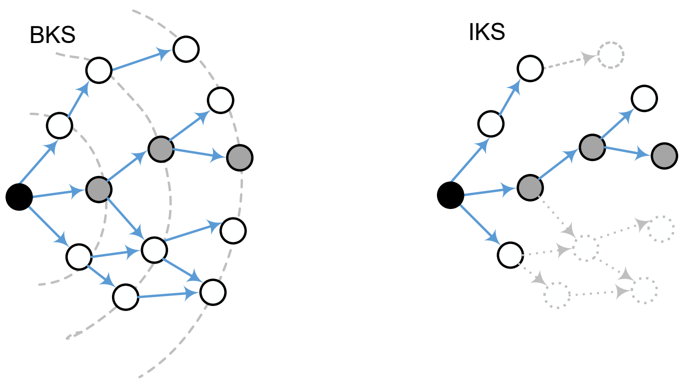

Fig 3. An example of finding the first answer by BKS and IKS methods.

where $|Q|$ shows the number of queried keywords and α represents a fixed parameter that is set to 2 in the experiments. As a node can receive information from multiple seed nodes and its prestige can be reduced by receiving new information, the prestige has been defined by means of a max function. The existence of $\alpha^{Level(nd)}$ in the denominator of (7) results in an exponential decrease in the prestige score with increasing the level of spread. It also controls the spreading level and prevents long progress towards very remote nodes. In order to have an efficient implementation, the level of nodes is saved during the query process. The initial level of any node is set to zero. This value is updated for any node n_d after receiving new information from an active node n_c by means of (8).

$$Level(n_d) = max(Level(n_d), Level(n_c + 1)) \quad (8)$$

The details of query processing by IKS is similar to that of the baseline method (BKS). In IKS, the next node is selected for activation from the unique priority queue and the new receiver nodes are added to the same queue. An answer is detected in a detector node and it is retrieved by executing a backward search started at the detector node. After retrieving an answer, the process to find the other answers continues with the existing nodes in the priority queue until k answers are retrieved or all nodes with a level lower than the retrieval depth are activated at least once. The pseudo-code of IKS is shown in Algorithm 1. The function ReduceAnswer in line 18 of the algorithm is used to convert an answer to a minimal one by eliminating its additional nodes.

To clarify the IKS model, consider the graph in Fig 2. The processing of query Q={IR,AI,ML,PL} on this graph is initialized by a queue with n_1(seed node) as its first entry. Node n_1 is the first node to activate. The HList of n_1 is sent to n_2 and n_3. After updating the HLists of receiver nodes (n_2 and n_3), their level is updated to 1 and their prestige score is calculated. Then, they are added to the priority queue. A node in the priority queue with the highest prestige is selected as the next active node. Therefore, n_2 is selected. The content of priority queue after activation of a sequence of nodes has been presented in Fig 4. Each entry shows the Id and prestige of a node. Based on this figure, node n_4 is activated earlier than node n_3 even though it is located at a greater distance from the seed node n_1.

This activation leads to the early detection of an answer in node n_5. It should be noted that the query coverage is checked after receiving new information in a node.

Due to the lack of a level order in the activation of nodes, the possibility of displacement of answers in the result list of IKS is more than that of BKS. However, in all cases, Theorem 2 is maintained.

Theorem 2. The distance between each pair of nodes in an answer produced by Informed keyword search is at most $log_{\alpha}|Q|$ larger than the length of the corresponding optimal path ($|Q|$ denotes the number of query keywords).

Proof: Suppose an optimal path from node s to node t with the length of d that covers the set of keywords Q={q_1,q_2,...,$q_{|Q|}$}. Without loss of generality, let the seed node s covers the queried keyword q_1 and the destination node t covers the queried keyword $q_{|Q|}$. It should be noted that the two nodes s and t certainly cover at least one unique keyword of the query. The worst-case in detecting the optimal path occurs when there is no positive sign of the presence of an answer on this path before reaching the last node. Such a situation is shown in the upper path in Fig 5. Node y on this path has the worst situation to expand.

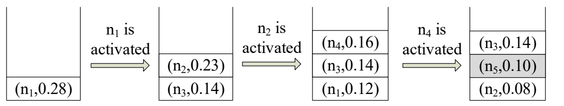

Fig 4. The content of priority queue during the processing of Q={IR,AI,ML,PL} by IKS (α=2).

Input: the input graph G; the query $Q = \{q_1, q_2, \dots, q_{|Q|}\}$; the number of answers k; the retrieval depth r.
Output: the set of top-k ordered answers.

```
1.  $S \leftarrow$ seed nodes of G related to Q
2.  **for** each $n_i \in S$ in descending order of $prestige(n_i|Q)$ do   //equation (7)
3.      **if** (Q ⊆ HList(n_i)) **then**
4.          Answers ← A, print(A)
5.          **if** (#Answers = k) **then return** Answers
6.      **else** $queue \leftarrow$ enqueue(n_i)
7.  **end for**
8.  **while** ($queue \neq \emptyset$) **do**
9.      $n_{act} \leftarrow$dequeue($queue$)
10.     **if** ($Level(n_{act}) > r$) **then continue**
11.     $\Gamma(n_{act}) \leftarrow$ direct neighbors of $n_{act}$
12.     send $HList(n_{act})$ to $\Gamma(n_{act})$
13.     **for** each $n_j \in \Gamma(n_{act})$ **do**
14.         **update** $HList(n_j)$ and insert new entries
15.         $Level(n_j) \leftarrow max\big(Level(n_j), Level(n_{act} + 1)\big)$
16.         **if** ($Q \subseteq HList(n_j)$) **then**
17.             $A \leftarrow$retrieve the answer detected by $n_j$ by a backward manner
18.             $A \leftarrow$ReduceAnswer(A)
19.             $Answers \leftarrow A$, print(A)
20.             **update** $HList(n_j)$ and delete used entries
21.             **if** ($\#Answers = k$) **then return** $Answers$
22.         **else** $queue \leftarrow$enqueue($\boldsymbol{n_j}$) in descending order of $prestige(n_j|Q)$  //equation (8)
23.     **end for**
24. **end while**
```

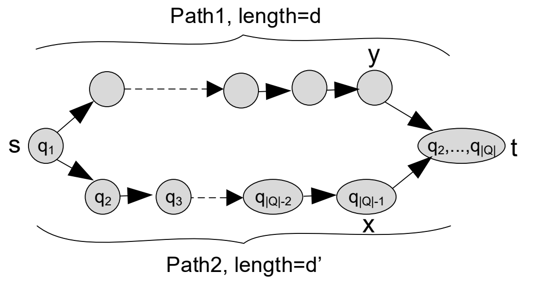

Fig 5. Two traversed paths in searching keyword set Q={q_1,q_2,...,q_n} using Informed Keyword Search.

The lower path in Fig 5 shows an alternative path between two nodes s and t covering all the queried keywords. This path is the most deceptive path between two nodes *s* and *t*. By spreading information from each of the two nodes x and *y*, an answer will be detected. Path2 in Fig 5 with greater length will be traversed earlier than the optimal path, if node x will be activated before node *y*. This happens when:

$$Prestige(y|Q) < Prestige(x|Q) \Rightarrow \frac{1+CR_N(q_1,s)}{(|Q|+1)\times\alpha^{d-1}} < \frac{(|Q|-1)+\prod_{i=1}^{|Q|-1} \underset{HList(x,q_i|Q)}{max} CR_N(q_i,x)}{(|Q|+1)\times\alpha^{d'-1}}$$

$$.\Rightarrow \ \alpha^{d'-d} < \frac{(|Q|-1)+\prod_{i=1}^{|Q|-1} \underset{HList(x,q_i|Q)}{max} CR_N(q_i,x)}{1+CR(q_1,s)}$$

$$\xRightarrow[\ ]{\substack{\forall q_i,n_d \\ 0\le CR_N(q_i,n_d)\le 1}} d' < d + log_\alpha|Q|$$

It means that if $d'<d+log_\alpha|Q|$, the wrong path will be traversed earlier than the optimal path. As this path is the most deceptive one, the maximum length of a wrong path is at most $log\alpha|Q|$ larger than the length of the optimal path.

V. EXPERIMENTS

1.Datasets and Queries

Coffman and Weaver [1, 38] developed an evaluation framework for testing the effectiveness of keyword search systems over data graphs. This framework consists of three real-world data sets: Mondial, Wikipedia, and IMDb. IMDb and Wikipedia have relatively large blocks of text as they describe quotes from movies and contents of pages, respectively. The text of Mondial is restricted to the names of countries, mountains, etc., but its graph has a complex structure in comparison to the other two datasets. Some information about the datasets is listed in Table 1.

In the evaluation framework, fifty queries with their exact relevant answers have been provided for each of these datasets. The average number of keywords per query is 2.91, while there are some queries having more than five keywords. The average number of relevant answers per query is 4.49. An answer is in the form of a single node or a connected sub-graph consisting of several nodes.

2. System Implementation

In the proposed work, any dataset of the evaluation framework has been modeled as a graph and the text has been only associated with its nodes. The text of each node contains some attributes including name, title, content, and any other information about the related entity. Due to the different importance levels of attributes, a weight has been calculated for each attribute. The attributes weight has been applied in determining the content-based relevance score of answers. Any node in the graph has a unique search Id. These Ids have been used to build an inverted index to the keywords and nodes in the graph. Stemming and stop-word removal have been applied to the text of nodes before indexing them.

Table 1. Characteristics of the evaluation datasets.

Dataset	$\|N\|$[a]	$\|E\|$[b]	$\|T\|$[c]
Mondial	17000	56000	12000
Wikipedia	206000	785000	750000
IMDb	1673000	6075000	1748000

[a]Nodes. [b]Edges. [c]Unique keywords.

Each entry of the inverted index is a stemmed keyword which points to a list of nodes containing it; this list is ordered by the content-based relevance score of nodes to the keyword. Using the inverted index, the nodes covering any queried keyword q_i could be efficiently retrieved.

The BKS and IKS algorithms have been implemented on Java using the JGraphT library which has been designed to scale to millions of vertices and edges. The retrieved results by the implemented systems have been shown in the same manner as in [38] to make them comparable with the results of the state of the art systems reported in [1, 38]. To evaluate the precision of the systems, three IR metrics of MAP (Mean Average Precision), top-1 (the accuracy of answers according to the first correct answer) and MRR (Mean Reciprocal Rank) have been used.

VI. EXPERIMENTAL RESULTS

The MAPs measured across various search methods and datasets are shown in Fig 6. The value of k (the number of retrieved answers) in this experiment was set to 100, although this value is very large for a lots of tested queries.

BANKS [13] and Bidirectional [14] are cluster-based search methods that rank answers using a notion of proximity coupled with a notion of the prestige of nodes based on in-links. DPBF [4] is a subgraph-based search system and Dup-free is a Lawler-based one. Both of these systems rank answers based on Steiner-tree semantic. Ranking answers solely based on their structural characteristics makes these systems unable to distinguish answers with similar structures but different semantics. The Effective method [8] uses IR scoring schemes coupled with proximity weighting to rank answers.

The comparison results in Fig 6 show that DBPF achieved the best performance on the Mondial dataset while it failed over the Wikipedia dataset. This method scores answers based on the semantics that: “if a relation has more neighbors, an edge that is incident on it reflects a weaker relationship between tuples” [4]. This belief was effective on the Mondial dataset which consists of a large number of tables with complex relationships, but it failed on the Wikipedia dataset which is a low dense graph. The MAP of this system on the IMDb dataset is not reported because of Memory exceptions. Most of the queries on Wikipedia have the answers with the same structure. Therefore, the results of the proximity-based systems on Wikipedia are not satisfactory. The Effective system also had a disappointing result on Wikipedia. This is due to the underlying property of IR-style ranking schemes: they prefer larger answers that contain additional instances of the queried keywords instead of smaller answers that satisfy the query [37]. The low MAP of Dup-Free system is related to removing the duplicate answers from the result set. However, the duplicate answers usually reflect different semantics, and the presence of all of them in the result set is essential. Fig 6 shows that the MAPs of the proposed systems without using a separate ranking phase are comparable with the MAPs of other state-of-the-art systems. The use of content-based scoring with preserving the structural priorities makes our systems more effective than some examined systems. As was expected, BKS is more effective than IKS over all datasets because of its exhaustive search on the data.

Fig 7 shows the MRR of systems for queries targeting exactly one relevant tuple (node). Such queries are the most common type of queries posed to existing search engines. Twenty queries on Mondial and IMDb datasets and fifteen queries on Wikipedia dataset are of this type. As the initial nodes are scored similarly in the two proposed systems, they get the same MRR in this experiment. Fig 7 shows the proper quality of our systems in comparison with the other systems.

To have a comparison in equal situations, the retrieved results by the proposed method are scored and ranked based on Eq. (5). It is because that the MAPs of the other systems have been reported based on the ranked results.

Fig 8 shows the MAPs of BKS and IKS after ranking answers using the proposed scoring function (presented in section 3.1) in comparison to the MAPs of the state-of-the-art systems. The superiority of the proposed systems shows the effectiveness of employed factors (the number of nodes, the diameter of the subgraph, and the pivoted normalization weighting) in the scoring of answers.

In the following experiment, the impact of retrieval volume on the MAP of systems before the ranking of results is examined. The value of k (the number of top answers) varied from 1 to 100, and the MAPs of three datasets have been reported in Fig 9. The diagram shows that both of the proposed systems can identify all the detectable relevant answers within the first 50 retrieved answers. This result is important because it shows that retrieving a limited number of answers is sufficient to be considered in the ranking phase. As evidenced by the diagrams, the MAP trends of the systems are relatively similar. BKS reasonably outperforms IKS in terms of MAP because of its more regularity in retrieving answers.

In Fig 10, the impact of retrieval depth (r) on the effectiveness of the proposed systems has been examined. In this experiment, 100 answers were retrieved for each query. Not surprisingly, effectiveness was improved by increasing the retrieval depth until all of the answers were retrieved. As the smaller r leads to lower execution time, the retrieval depth of Mondial, Wikipedia, and IMDb were set to 5, 3, and 2, respectively, in all the experiments.

One of the metrics to evaluate the performance of systems is execution time. We reported the total execution time of the proposed systems for all the queries by varying the value of k in Fig 11. It should be noted that the experiments were run on an Intel(R) Core(TM) i7 QM 2.50 GHz computer with 12GB of RAM. The outperformance of IKS over BKS in terms of execution time is evident in this experiment.

The impact of retrieval depth on the total execution time of queries has been shown in Fig 12. In this experiment, 100 answers were retrieved in each depth. Advancing up to higher retrieval depths provides the possibility of retrieving answers with more radiuses. The exponential increase trend in execution times of queries when the retrieval depth is increased is observed in the diagrams of Fig 12. Nevertheless, based on the results shown in Fig 10, the effectiveness of systems over large datasets does not improve by increasing the retrieval depth to more than 2.

The second metric for performance evaluation is response time which is defined as the time elapsed from issuing the query until the first answer is returned by the system. The average

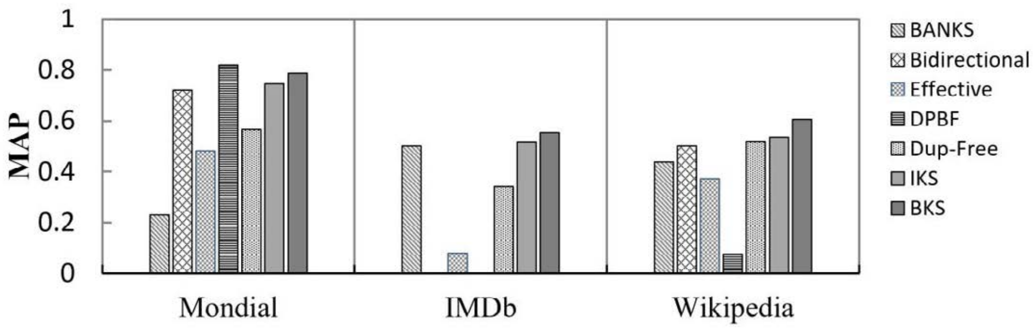

Fig 6. The comparison on MAP of the proposed systems before ranking results with MAP of the state-of-the-art systems.

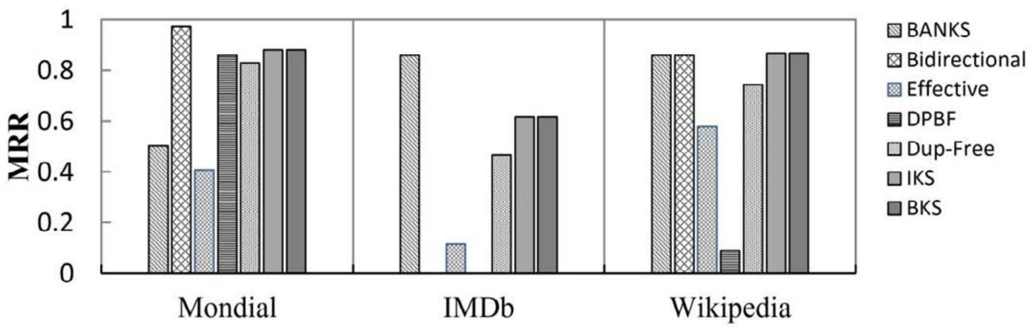

Fig 7. The comparison on MRR of different systems for queries which are associated with single-node answers.

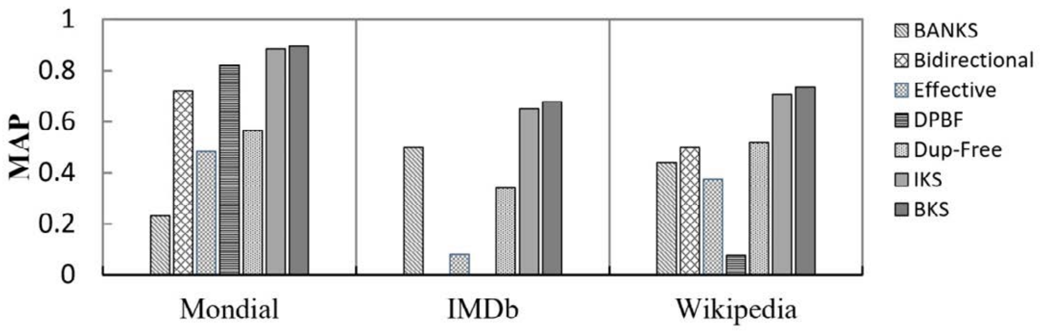

Fig 8. The comparison on MAP of the proposed systems after ranking results with MAP of the state-of-the-art systems.

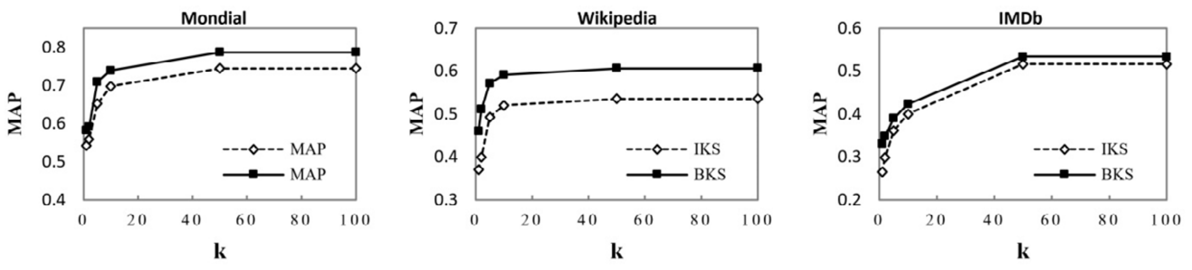

Fig 9. The overall search effectiveness of the proposed systems by varying k.

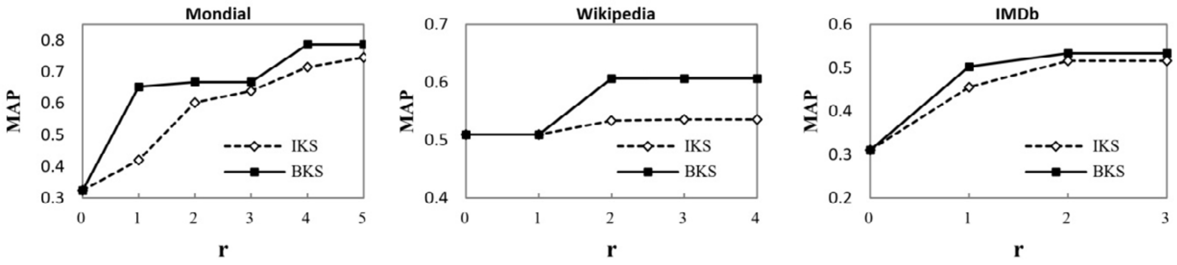

Fig 10. The overall search effectiveness of the proposed systems by varying retrieval depth.

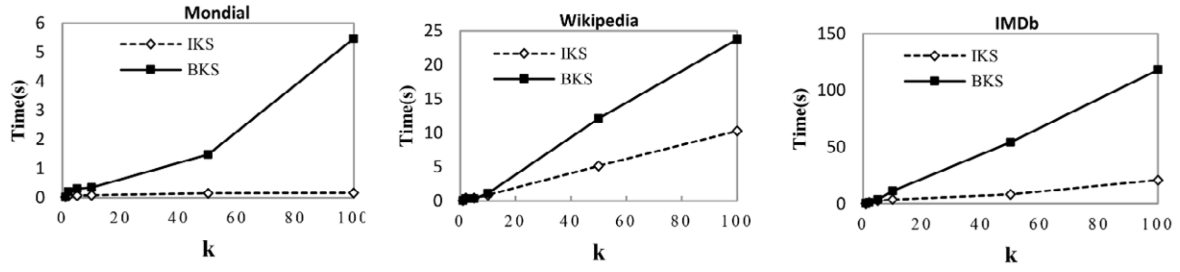

Fig 11. The total execution time of the proposed systems by varying k.

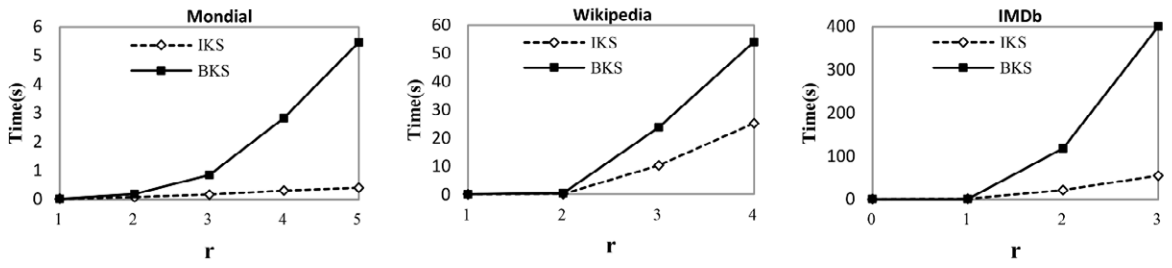

Fig 12. The total execution time of the proposed systems by varying retrieval depth.

response times of the proposed systems over all queries for the three datasets are shown in Table 2. In this table, P@k represents mean precision across all queries where the number of top answers is limited to k. The data provided in Table 2 show "interactive" response time (on the order of a few seconds) [1] for each system. All of the systems are able to deliver the first answer of queries in less than one second, indicating that the proposed systems are practical.

VII. DISCUSSION AND CONCLUSION

In this paper, we proposed two heuristic algorithms based on the information spreading technique to retrieve relevant answers in response to interactive keyword queries. In Blind Keyword Search, the information is spread in a regular manner to all in-range nodes while in Informed Keyword Search the information is spread to the nodes having more potential to be present in an answer. IKS substantially improved the performance of BKS with just a minor impact on its Mean Average Precision (i.e., quality of answers).

Table 2. A comparison between BKS and IKS systems.

	IKS			BKS		
Dataset	Resp. (s)[a]	Exec. (s)[b]	*P*@1	Resp. (s)	Exec. (s)	*P*@1
Mondial	0.021	0.1646	0.52	0.0152	5.4692	0.56
Wikipedia	0.0048	10.272	0.48	0.0071	23.747	0.51
IMDb	0.0644	20.876	0.22	0.591	118.074	0.24

[a]Mean response time (in second).
[b]Mean execution time to retrieve 100 answers (in second).

The experimental results on a wide range of queries show that the effectiveness of the proposed systems is comparable with that of state-of-the-art systems utilizing a ranking phase to sort retrieved answers. This achievement is the result of the proper use of conceptual factors during the search process. With regard to efficiency, the response time of the proposed systems is very lower than their execution time on various queries because the answers are presented immediately after they are retrieved. Another advantage of IKS and BKS is that their processing time is independent of the size of the base graph. This leads to relatively similar average processing times of the proposed systems over two datasets of Mondial and Wikipedia though they have different sizes. However, the dependence of the processing times on the branching factor of the graphs makes these systems efficient just on sparse graphs.

Among the algorithms proposed in this paper, BKS is easily extensible to be used in distributed environments. To this end, an auxiliary database would be used to store the content of nodes. The graph structure and the Id of nodes would be maintained in the main memory and the content of nodes would be saved in the database. The required information would be fetched to the main memory after the first activation of a node. To distribute keyword search, the structure of the graph is saved on all the servers and they are connected to the database to fetch the needed information. Each server is responsible to retrieve a group of answers related to a specified seed node. The data of the nodes close to some seed nodes may be fetched several times from the database by different servers. It is not necessary for each server to inform the other servers about the changes of common nodes because each group of answers related to a seed node would be retrieved independently from the other answers. To execute BKS, the steps of the algorithm should be simultaneously started by all the servers to preserve the approximate order of answers. In future works, the distributed form of keyword search will be followed.

REFERENCES

1. J. Coffman, A.C. Weaver, An Empirical Performance Evaluation of Relational Keyword Search Techniques, IEEE Transactions on Knowledge and Data Engineering, 26 (2014) 30-42.
2. J. Park, S.-g. Lee, Keyword search in relational databases, Knowl Inf Syst, 26 (2011) 175-193.
3. A. Ghanbarpour, H. Naderi, Survey on Ranking Functions in Keyword Search over Graph-Structured Data, Journal of Universal Computer Science, 25 (2019) 361-389.
4. B. Ding, J.X. Yu, S. Wang, L. Qin, X. Zhang, X. Lin, Finding Top-k Min-Cost Connected Trees in Databases, in: 23rd International Conference on Data Engineering, IEEE, 2007, pp. 836-845.
5. H. He, H. Wang, J. Yang, P.S. Yu, BLINKS: ranked keyword searches on graphs, in: Proceedings of the 2007 ACM SIGMOD international conference on Management of data, ACM, Beijing, China, 2007, pp. 305-316.
6. M. Kargar, A. An, X. Yu, Efficient Duplication Free and Minimal Keyword Search in Graphs, IEEE Transactions on Knowledge and Data Engineering, 26 (2013) 1657 - 1669
7. M. Kargar, A. An, Keyword search in graphs: finding r-cliques, Proceedings of the VLDB Endowment, 4 (2011) 681-692.
8. F. Liu, C. Yu, W. Meng, A. Chowdhury, Effective keyword search in relational databases, in: Proceedings of the 2006 ACM SIGMOD international conference on Management of data, ACM, Chicago, USA, 2006, pp. 563-574.
9. M. Miao, J. Wang, S. Wen, J. Ma, Publicly verifiable database scheme with efficient keyword search, Information Sciences, 475 (2019) 18-28.
10. V. Hristidis, L. Gravano, Y. Papakonstantinou, Efficient IR-style keyword search over relational databases, in: Proceedings of the 29th international conference on Very large data bases, VLDB Endowment, Berlin, Germany, 2003, pp. 850-861.
11. C.-S. Park, S. Lim, Efficient processing of keyword queries over graph databases for finding effective answers, Information Processing & Management, 51 (2015) 42-57.
12. X. Yu, Z. Yu, Y. Liu, H. Shi, CI-Rank: Collective importance ranking for keyword search in databases, Information Sciences, 384 (2017) 1-20.
13. A. Hulgeri, C. Nakhe, Keyword Searching and Browsing in Databases using BANKS, in: Proceedings of the 18th International Conference on Data Engineering, IEEE Computer Society, 2002, pp. 431-440.
14. V. Kacholia, S. Pandit, S. Chakrabarti, S. Sudarshan, R. Desai, H. Karambelkar, Bidirectional expansion for keyword search on graph databases, in: Proceedings of the 31st international conference on Very large data bases, VLDB Endowment, Trondheim, Norway, 2005, pp. 505-516.
15. J. Vivekavardhan, A.S. Chakravarthy, P. Ramesh, Search Engines and Meta Search Engines Great Search for Knowledge: A Frame Work on Keyword Search for Information Retrieval, in: S.C. Satapathy, K.S. Raju, K. Shyamala, D.R. Krishna, M.N. Favorskaya (Eds.) Advances in Decision Sciences, Image Processing, Security and Computer Vision, Springer International Publishing, Cham, 2020, pp. 435-443.
16. Y. Mass, Y. Sagiv, Language models for keyword search over data graphs, in: Proceedings of the 5th ACM international conference on Web search and data mining, ACM, Seattle, Washington, USA, 2012, pp. 363-372.
17. J.I. Lopez-Veyna, V.J. Sosa-Sosa, I. Lopez-Arevalo, A low redundancy strategy for keyword search in structured and semi-structured data, Information Sciences, 288 (2014) 135-152.
18. S. Najafi, F. Soleimanian Gharehchopogh, A New Hybrid Method for Web Pages Ranking in Search Engines, Journal of Advances in Computer Engineering and Technology, 5 (2019) 233-244.
19. A. Sela, I. Ben-Gal, Information spread in the age of the internet, in: 2014 IEEE 28th Convention of Electrical & Electronics Engineers in Israel (IEEEI), 2014, pp. 1-4.
20. C. Liu, Z.-K. Zhang, Information spreading on dynamic social networks, Communications in Nonlinear Science and Numerical Simulation, 19 (2014) 896-904.
21. S. Bergamaschi, F. Guerra, M. Interlandi, R. Trillo-Lado, Y. Velegrakis, QUEST: a keyword search system for relational data based on semantic and machine learning techniques, Proceedings of the VLDB Endowment, 6 (2013) 1222-1225.
22. S. Bergamaschi, F. Guerra, M. Interlandi, R. Trillo-Lado, Y. Velegrakis, Combining user and database perspective for solving keyword queries over relational databases, Information Systems, 55 (2016) 1-19.
23. T.N. Le, Z. Bao, T.W. Ling, Exploiting semantics for XML keyword search, Data & Knowledge Engineering, 99 (2015) 105-125.
24. P. Dayananda, C.N. Sowmyarani, Review on Keyword Search and Ranking Techniques for Semi-Structured Data, in: D.L. Miltiadis, D. Linda, V. Anna (Eds.) Knowledge-Intensive Economies and Opportunities for Social, Organizational, and Technological Growth, IGI Global, Hershey, PA, USA, 2019, pp. 248-270.
25. M. Rihany, Z. Kedad, S. Lopes, A Keyword Search Approach for Semantic Web Data, in: E. Métais,

F. Meziane, S. Vadera, V. Sugumaran, M. Saraee (Eds.) Natural Language Processing and Information Systems, Springer International Publishing, Cham, 2019, pp. 131-143.

26. D. Wang, L. Zou, D. Zhao, Top-k queries on RDF graphs, Information Sciences, 316 (2015) 201-217.

27. S.B. Sinha, X. Lu, D. Theodoratos, Personalized Keyword Search on Large RDF Graphs based on Pattern Graph Similarity, in: Proceedings of the 22nd International Database Engineering & Applications Symposium, ACM, Villa San Giovanni, Italy, 2018, pp. 12-21.

28. Y. Yuan, G. Wang, L. Chen, H. Wang, Efficient Keyword Search on Uncertain Graph Data, IEEE Transactions on Knowledge and Data Engineering, 25 (2013) 2767-2779.

29. S. Kim, W. Lee, N.R. Arora, T.-C. Jo, S.-H. Kang, Retrieving keyworded subgraphs with graph ranking score, Expert Systems with Applications, 39 (2012) 4647-4656.

30. Y. Yang, D. Agrawal, H.V. Jagadish, A.K.H. Tung, S. Wu, An Efficient Parallel Keyword Search Engine on Knowledge Graphs, in: 2019 IEEE 35th International Conference on Data Engineering (ICDE), 2019, pp. 338-349.

31. G. Li, B.C. Ooi, J. Feng, J. Wang, L. Zhou, EASE: an effective 3-in-1 keyword search method for unstructured, semi-structured and structured data, in: Proceedings of the 2008 ACM SIGMOD international conference on Management of data, ACM, Vancouver, Canada, 2008, pp. 903-914.

32. R.D. Virgilio, P. Cappellari, M. Miscione, Cluster-Based Exploration for Effective Keyword Search over Semantic Datasets, in: A.F. Laender, S. Castano, U. Dayal, F. Casati, J. de Oliveira (Eds.) Conceptual Modeling - ER 2009, Springer Berlin Heidelberg, 2009, pp. 205-218.

33. K. Nguyen, J. Cao, Top-K data source selection for keyword queries over multiple XML data sources, Journal of Information Science, 38 (2012) 156-175.

34. Y. Mass, Y. Sagiv, Virtual Documents and Answer Priors in Keyword Search over Data Graphs, in: EDBT/ICDT Workshops, 2016.

35. Y. Pan, Y. Wu, ROU: advanced keyword search on graph, in: Proceedings of the 22nd ACM international Conference on information and knowledge management, ACM, San Francisco, California, USA, 2013, pp. 1625-1630.

36. B. Kimelfeld, Y. Sagiv, Finding and approximating top-k answers in keyword proximity search, in: Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, ACM, Chicago, IL, USA, 2006, pp. 173-182.

37. Y. Hao, H. Cao, Y. Qi, C. Hu, S. Brahma, J. Han, Efficient keyword search on graphs using mapreduce, in: Big Data (Big Data), 2015 IEEE International Conference on, IEEE, 2015, pp. 2871-2873.

38. J. Coffman, A.C. Weaver, A framework for evaluating database keyword search strategies, in: Proceedings of the 19th ACM international conference on Information and knowledge management, ACM, Toronto, ON, Canada, 2010, pp. 729-738.