Distributed Information Retrieval Based On Metaheuristic Search and Query Expansion

Alia Karim Abdul Hassan
Computer Science Department,
University of Technology/Baghdad, Iraq
hassanalia2000@yahoo.com

Mustafa Jasim Hadi
Computer Science Department,
University of Technology/Baghdad, Iraq
mustafa_awadi@yahoo.com

Abstract— *Distributed information retrieval (DIR) is a model enables a user to access many searchable databases reside in different locations. DIR is more complex than the centralized information retrieval (IR). It requires addressing two significant additional problems that are the resource selection and the results merging. Many techniques for addressing the two problems have been published in the literature. However, they still have a negative impact on retrieving quality and response time. This paper aims to improve the DIR efficiency through using a meta-heuristic algorithm and improving the result quality through a query expansion. The algorithm has been strengthened using the nearest neighbor graph in order to improve the search performance. The performance in the proposed system outperforms the one in the traditional system in a rate from 6% to 9% while reduces the latency in an approximate rate from 0.047 to 0.134 second for each query.*

Keywords—Distributed Information Retrieval; Resource selection; Results merging; Meta-heuristic; Query Expansion.

I. INTRODUCTION

Information retrieval (IR) is the field of computer science that deals with the storage, search, and retrieve the information such as texts, Web pages, images, and videos. IR systems are used in a wide range of application areas such as web search, digital library search, social search, recommender system, etc. The major concern of IR is to find relevant information (documents) that satisfy the user need [1]. Nearly all IR systems today rely on a data structure called an inverted index by which a query term matches an index term associated with a list of documents that contain it. After the extraction of the documents from the inverted index, the relevant documents are retrieved depending on the score of the query-document similarity.

The growth of the Internet and digital libraries increased the need to address the IR in a distributed environment. The sources of information have become varied dramatically and their contents cannot be explored and indexed by a centralized IR system [2]. Distributed Information Retrieval (DIR) has received much attention in recent years. A simple DIR system is consisting of collection servers and a broker. When a user submits a query to the broker, the broker will propagate that query to a subset of collection servers that are carefully selected to satisfy the user need. After the query is processed by the selected collection servers, a ranked list from each one is returned to the broker. Finally, the broker merges the ranked lists into a unified ranked list and passing their contents to the user [3]. DIR is more complex than the centralized information retrieval (IR). It requires addressing two significant additional problems that are the resource selection and the results merging [4]. The motivation of this paper is to improve both the solution quality and response time. The response time is improved using a stochastic-based bio-inspired algorithm called Artificial Bee Colony (ABC) algorithm, which falls under the umbrella of swarm intelligence. The algorithm is modified to suit the required search mechanism. The neighbor documents are constructed using the nearest neighbor graph that improves ABC performance. The solution quality is improved using a query expansion process. The query is expanded by synonyms extracted automatically by a proposed mechanism.

The rest of the paper is organized as follows: Section II reviews the brief description of the information retrieval and the major problems in the distributed information retrieval. Section III exhibits the mechanism of artificial bee colony as a meta-heuristic algorithm. Section IV descripts how to construct a nearest neighbor graph to improve the meta-heuristic search. Section V reviews the important related works. The reminder sections from VI to VIII exhibit the proposed system, experimental results, and conclusions.

II. INFORMATION RETRIEVAL

Information retrieval (IR) is a process to represent, store, organize and search the information items. Information must

be structured in some manner that ensures the relevant information retrieval. Vector space model (VSM) is most widely used in IR to retrieve the documents. In VSM the documents and queries are stored in weights vectors. The weight vector of a document d is defined as $\vec{d} = (w_{d,1}, w_{d,2}, \ldots, w_{d,n})$ while the weight vector of a query q is defined as $\vec{q} = (w_{q,1}, w_{q,2}, \ldots, w_{q,n})$. The weight for a term is calculated using *tf-idf* weighting scheme, tf refers to the term frequency and idf refers to the inverse document frequency. Cosine similarity metric is one of the most popular similarity measures that finds the normalized dot product of the two above vectors as follows[5]:

$$cos\ (q,d) = \frac{\vec{q}.\vec{d}}{|\vec{q}|.|\vec{d}|} = \frac{\sum_{t=1}^{n} w_{q,t} \cdot w_{d,t}}{\sqrt{\sum_{t=1}^{n} w_{q,t}{}^2} \cdot \sqrt{\sum_{t=1}^{n} w_{d,t}{}^2}} \quad (1)$$

Where $w_{q,t} = tf_{q,t} \times log\ (N/df_t)$ and $w_{d,t} = tf_{d,t} \times log\ (N/df_t)$. The expression $log\ (N/df_t)$ refers to *idf, N* is the number of documents in the document collection and df is the number of documents in which the term appears .

The most common traditional search approach in IR applications is inverted-index based search. Through using the inverted index, the search complexity of query-document with non-zero similarity is reduced at a phenomenal rate. However, the inverted index file may become untreatable for the large-scale databases or the web. Metaheuristic approaches such as the swarm intelligence can get a response time with a polynomial rate on a higher computation scale [6], [7]. Improvement of the quality of IR using the word sense disambiguation (WSD) was subject of controversy among a lot of authors. However, the recent researches such as in [8],[9] pay tribute to the positive role for using the WSD to improve the IR systems. Query expansion is one of the means to improve the IR using the WSD. IR systems vary between being centralized or distributed.

Distributed information retrieval (DIR) is a model in which a user accesses many searchable documents reside in a set of document collection servers [10]. DIR is more complex than the centralized IR. It requires addressing two significant problems are the resource selection and the results merging.

A. Resource selection

Resource selection refers to finding a subset of document collections located on remote servers that have the most probability to contain the required information by a given query. This step is important to reduce the response time while trying to maintain the quality as much as possible [3]. When the broker receives a query, it forwards the query to the selected resources that use their local algorithms to rank the documents. Each selected resource returns a ranked list with scores to the broker. The broker normalizes the scores of the ranked lists for making them comparable in order to be unified in the next step, the results merging [11].

B. Results merging

The results merging step is the last phase of the DIR process. The individual ranked lists of the local collection servers are merged into a unified list that will be returned to the user submitting the query. Merging the ranked lists of individual IR systems is a complex problem because the diversity of individual collection statistics that leads to inconsistent servers scores [12].

III. ARTIFICIAL BEE COLONY ALGORITHM

Artificial Bee Colony (ABC) algorithm is a meta-heuristic bio-inspired algorithm falls under the umbrella of swarm intelligence. The ABC algorithm is introduced by Karaboga [13]. The main steps of the ABC algorithm are described as follows [14]:

Step 1: Initialize the population of random food sources and evaluate them.

Step 2: Produce new sources for the employed bees, evaluate them and apply the greedy selection process.

Step 3: Compute the probability values of the current sources to be used for the selection process by the onlookers.

Step 4: Produce new sources for the onlookers from selected sources, evaluate them and apply the greedy selection process.

Step 5: Determine the abandoned resources and send the scouts randomly in the search area for discovering alternative new food sources.

Step 6: Memorize the best food source achieved so far.

Step 7: If the termination condition is not met, return to step 2, otherwise stop.

In the initialization phase, the initial solutions are computed as follows:

$$x_{ij} = x_{min\ j} + rand\ [0, 1] * (x_{max\ j} - x_{min\ j}) \quad (2)$$

Where $i \in \{1,..,N_S\}$, $j \in \{1,..,D\}$, N_S is the number of food sources, and D is the number of optimized parameters, $x_{max\ j}$ and $x_{min\ j}$ refere to the upper and lower bounds for the dimension j. *rand[0, 1]* is a random number between [0, 1]. In the employed bees and onlooker bees phases, the new solutions v_{ij} are computed as follows:

$$v_{ij} = x_{ij} + \phi_{ij}\ (x_{ij} - x_{kj}) \quad (3)$$

Where $i,k \in \{1,..,N_S\}$ and $j \in \{1,..,D\}$ are randomly chosen indexes, ϕ_{ij} is a random number between [-1, 1]. The selection of the new solutions in the onlooker bees is depending on probability p_i that is computed as follows:

$$p_i = a \times \frac{f_i}{f_{max}} + b \quad (4)$$

Where (a+b=1), f_i is the fitness value of the food source *i*, f_{max} is the maximum fitness of food sources.

In the Scout bees phase, a new source v_{ij} is randomly generated instead of an abandoned one depending on the equation (4) [15].

IV. NEAREST NEIGHBOR GRAPH CONSTRUCTION

The artificial bee colony (ABC) algorithm is improved using the nearest neighbor graph. Given a vector space model $X_{n\times m}$ for a document collection, for all pairs of documents in X, w_{ij} is a value reflecting the similarity between two documents x_i and x_j . The most common similarity function is the cosine similarity $cos(x_i, x_j)$ just as in Eq. (1). The document similarity graph $G = (V, E, W)$ for a document collection is an undirected weighted graph in which V is a set of nodes and each node is mapped to a document $x_i \in X$, E is a set of edges refer to the document relationships, and W is the similarity weights. For each pair of documents x_i and x_j , there is an edge $e_{ij} \in E$ connects the respective nodes with weight w_{ij} equal to $cos(x_i, x_j)$. In order for the edges with low weights to be not included in the similarity graph, a threshold ε should be used. The threshold ε is a parameter to control the number of links in the graph. The higher the value of ε, the fewer the links can exist in the graph with high similarity scores. In other words, the $\varepsilon-$neighborhood document similarity graph G^{ε} is the document similarity graph $G = (V, E, W)$ in which each edge e_{ij} in the graph can connect two nodes (documents) with weight $w_{ij} = cos(x_i, x_j)$ if and only if $cos(x_i, x_j) \geq \varepsilon$ [16].

In this paper, the proposed system constructs one similarity graph G^{ε} for all the document collections in the system and preserves it in a single server for improving the neighboring search in the ABC algorithm at the query time.

V. RELATED WORKS

The related works in this paper are divided into two main sections:

A. Distributed information retrieval

Some methods to address the resource selection and results merging are shown as follows:

1. *The resource selection*: Paltoglou [12] provides an overview of a significant number of resource selection methods such as Glossary of Servers Server (GlOSS), Collection Retrieval Inference network (CORI), Cue Validity Variance (CVV), etc. The author shows that the CORI method is widely used as a baseline in the researches by a lot of authors (and the author himself) and it is more effective than both GlOSS and CVV.

In this paper, we just focus on CORI method. CORI is an algorithm developed by Callan et al. [17], [18], [4] to calculate the collection servers' scores. It considers that each collection server as a single giant document. Ranking the collection servers is similar to document ranking approaches used in the centralized IR systems [3]. It calculates belief scores of local collection servers based on a Bayesian inference network model with an adaptation of the Okapi term frequency normalization formula [19]. Let Q is a given query, S_i is a collection server, and t_k is a term occurs in S_i. The local collection servers are ranked based on the probability $p(Q|S_i)$ that refers to the belief that a query Q is satisfied by a local collection server S_i. The belief $p(t_k|S_i)$ indicates that the search term t_k contributes in finding the score of collection server S_i and is computed as follows [12], [20]:

$$p(t_k|S_i) = b + (1 - b) * R * I \quad (5)$$

Where:

$$R = \frac{df}{df+50+150*\frac{nt}{avg\_nt}},\ I = \frac{log(\frac{|C|+0.5}{sf})}{|C|+1.0}$$

Where df is the number of documents that contain term t_k within the collection server S_i, sf is the number of collection servers that contain term t_k, nt is the number of terms in S_i, avg_nt is the average nt, $|C|$ is the total number of collection servers and b is the default belief, the default value of b is 0.4. Finally, the belief $p\,(Q|S_i)$ for the total query terms is computed as follows:

$$p\,(Q|S_i) = \frac{\sum_{t_k \in Q}\ p\,(t_k|S_i)}{|Q|} \quad (6)$$

2. *The results merging*: many solutions for this issue are present in the literature. The simplest methods for addressing this problem are either the sequential combination of the results or the interleaved combination of the results in a Round Robin (RR) fashion. The RR merging method is used as a baseline for results merging by a lot of authors because of its simplicity and it introduces better results than the sequential combination method. The RR merging is defined as follows: given n ranked lists L_1, $L_2 \ldots L_n$, take the first result r_1 from each list L_i as the first n results, then, take the second result r_2 from each list as the next n results, and so on [21]. Another solution that also serves as a baseline is the CORI merging method that is associated with the CORI resource selection algorithm mentioned in advance. It is one of the most widely used methods due to its effectiveness and its simplicity [12]. The results merging algorithm is a combination of the original document score and collection selection score. It uses a simple heuristic method to normalize resultant scores [22]. The normalization of the collection and document scores is as follows [12]:

$$S_i' = \frac{S_i - S_{min}}{S_{max} - S_{min}} \quad (7)$$

$$D' = \frac{D - D_{min}}{D_{max} - D_{min}} \quad (8)$$

$$D'' = \frac{D' + 0.4 * D' * S_i'}{1.4} \quad (9)$$

Where S_i is the local collection score, S_{min} and S_{max} are the minimum and maximum collection server scores, S_i' is the normalized collection server score. D is the original relevance score of a document given by the local collection server, D_{min} and D_{max} are the maximum and minimum document scores form a local collection. D' is the document normalized score. Finally, D'' is the final document score that is normalized by dividing by 1.4.

Recently, Saoud and Kechid [20] present an approach that exploits the social profile to improve both the source selection and the result merging process. To simulate a real DIR system and provide the social information, they construct their own document collection using a social bookmarking system and others from the Web. The reported results show this method improves the relevance metrics in DIR systems.

In this paper the traditional algorithms for the resource selection and results merging is used for the comparison process. The CORI methods for both the resource selection and merging are used for the traditional DIR system. Whereas the selection algorithm in our proposed DIR system is conducted randomly using the ABC approach and the merging method is the RR merging method.

B. Centralized information retrieval

We focus on the intelligent methods inspired of swarm intelligence techniques that were previously applied to the centralized IR systems. Ramya and Shreedhara [23] offered in their paper a brief review on the application of swarm intelligence to centralized IR systems. They offer different swarm intelligence methods that aim to improve the search mechanism in the large-scale databases and the Web. Drias and Mosteghanemi [6] designed an algorithm called BSO-IR inspired from the Bees Swarm Optimization (BSO) algorithm to explore the prohibitive number of documents to obtain the most relevant results. They used CACM and RCV1 corpuses to test their experiments. The previous works published recently in [8], [9] present new techniques use the Artificial Bee Colony (ABC) approach to address the problem of using the Word sense Disambiguation (WSD) in the centralized IR systems. The original ABC algorithm was modified to suit the required solutions.

VI. PDEABC SYSTEM

In the distributed information retrieval systems, the resource selection is a mechanism to improve efficiency while the results merging is a mechanism to improve result quality. However, many challenges related to the response time and result quality are still under discussion among a lot of authors. Although the resource selection mechanism improves efficiency, but it may harm the result quality. Also, the results merging mechanism improves result quality, but it may harm the system efficiency due to the complex online computations. This is why we need an innovative tool to overcome the weaknesses in the efficiency and the result quality. PDEABC system is proposed to address these two problems based on ABC algorithm and query expansion. Where P refers to proposed, D refers to distributed, and E refers to the expansion. The CORI methods for both the resource selection and merging are used for the traditional distributed IR system. Whereas the selection algorithm in PDEABC system is conducted randomly depending on ABC algorithm and the merging method is simply the RR merging method. PDEABC system architecture consists of three main components as follows:

1. The documentary database (offline part): Let DS is a set of document collections used in the proposed system, where DS_k $(1 \leq k \leq m)$. Each collection is stored on a server S_k as local Vector Space Model (VSM) after the preprocessing and the term weighting have been achieved. Also, there is an additional server used to store the similarity graph, G^{ε}.
2. The query vector (online part): Let Q is a query submitted to the PDEABC system. The query should convert to query vector after the preprocessing and the term weighting have been achieved. Also associate each query word with list of different synonyms extracted from the WordNet using a proposed method. Initially all the synonyms have equal evaluations (zero fitnesses).
3. The search and matching mechanism.`

In more details, PDEABC system consists of three proposed algorithms called "PNWSD Algorithm", "MABC-PCS Algorithm", and "Expand Query & Matching Algorithm" respectively. Where PNWSD refers to a proposed word sense disambiguation (WSD) that depends on the knowledge-based resource (WordNet resource), this algorithm detects the best sense in a manner as if we used a miniature traditional IR system based on the inverted index. This implies a presence of miniature inverted index used for search process and a miniature VSM used for similarity calculation process. The miniature inverted index indexes all the words, except stopwords, in the glosses that correspond to all senses related to all the given ambiguous words and associate those to information about their glosses and their frequencies. Whereas the miniature VSM is a statistical model extracted from WordNet for all the ambiguous words' glosses and used to compute the cosine similarity between its glosses vectors and the vector that models the context that contains the ambiguous words. MABC-PCS refers to a modified ABC algorithm and a proposed synonym selection based on a given corpus as a resource (NPL corpus), which is raw data divided and modeled in more than one local VSM. Each local VSM is a statistical model extracted from corresponding local documents. They used for computing the similarity between the query and documents for retrieving process as well as the disambiguation as in the miniature VSM, but the glosses here are information from the corresponding documents. The three algorithms are presented as follows:

Algorithm 1: PNWSD Algorithm

Input:$Q = \{w_{q,i} | i = 1..n\}$, $w_{q,i}^{G}$=$\{w_{q,i}^{g} | g = 1..k\}$, $w_{q,i}^{g}$=$\{w_{q,i}^{g,j} | j = 1..m\}$ /* $w_{q,i}$ is a query word, $w_{q,i}^{g}$ is a gloss for $w_{q,i}$ within a set of glosses $w_{q,i}^{G}$ associated with $w_{q,i}$, the glosses are extracted from WordNet , all the keywords in the glosses are stored in a miniature inverted index for speedup the processing. */

Output: $S_q^{Synsets_{best}}$/* Best set of synonyms for each query word*/

Extracting Best Senses:

Begin

For i = 1 to n do

Get $w_{q,i}^{G}$ from the matching process between $w_{q,i}$ and all the glosses' keywords that are stored in the miniature inverted index.

Calculate the cosine similarity between the query words and $w_{q,i}^{g}$.

Rank the cosine similarities in decreasing order. Consider the first is the winner gloss and the finds the associated sense from the WordNet.

Extract the set of synonyms that are associated with the best sense using the WordNet

End for

End

After we get a set of synonyms for each query word using PNWSD algorithm, we try now to find just one of the synonyms that is the best among the others. This will be automatically based on the corpus using a proposed approach. The proposed approach is designed using the modified ABC (MABC). MABC is intended for two search processes, the first is to search the most relevant documents and the second is to search the best synonyms using the corpus as a resource. Thus we called this as MABC-PCS algorithm, where PCS refers to the proposed corpus-based synonym selection method.

Algorithm2: MABC-PCS Algorithm

Input: G^{ε}, Q, DS, $S_q^{Synsets_{best}}$, I_{max} .

Output: $Gdoc^{local_k}$, $S_q^{synonym_{best}}$=$\{w_{q,i}^{synonym_{best}} | i = 1..n\}$.

Search Mechanism:

Begin

Step1: Initialize the population: Set of random food sources (documents) scattered in all the servers S_k ($1 \le k \le m$).

Step2: Calculate the fitness values of food sources in the population.

Step3: Update the synonyms scores of query words.

Step4: cycle = 1

Step5: Repeat

5.1: Memorize $Gdoc^{global}$ and $Gdoc^{local_k}$ so far solutions. Let the neighbors of $Gdoc^{global}$ is $docs^{global}$.

5.2: Determine the neighbors $ni\_docs_i^{local_k}$ from G^{ε} of the chosen food sources $doc_i^{local_k}$ for the employed bees and evaluate them. Update the synonyms scores of query words.

5.3: Determine the neighbors ni_docs^{global} from $G^{\varepsilon}$of the chosen food sources $doc_i^{local_k}$ based on the probability for the onlooker bees and evaluate them. Update the synonyms scores of query words.

5.4: Produce the new food sources for the abandoned food sources by the scouts after scattering them to all the servers. Calculate their fitness values. Update the synonyms scores of query words.

5.5: cycle = cycle + 1.

Step6: Until cycle = I_{max} .

Step7: Calculate *tf-idf* weight for best synonyms.

End.

PDEABC system constructs one similarity graph G^{ε} for all the document collections in the system and preserves it in a single server for improving the neighboring search in at the query time. Initially the documents are randomly chosen in an integer interval from 1 to the total collections size (i.e. summation of sizes of all the collections). The fitness function in the search algorithm is the cosine similarity between the corresponding weighted term vectors of a query and a document. At each iteration, the new documents are evaluated and the synonyms scores are updated. The updating of the synonym scores depends on the assumption that each document can be considered as a gloss or a definition of that synonym and the query is the context where we need to find the suitable synonym. The synonym is then selected depending on the document that has the word match the query word synonym and offers the maximum similarity with the query. An update of the synonyms scores must be done after the existence of matching between the synonyms and current document words as well as if the new fitness is larger than the old one. The document has the best value at all, $Gdoc^{global}$, and the document has the best value within a local server S_k, $Gdoc^{local_k}$, are all memorized. The current document in the employed phase, $doc_i^{local_k}$, is replaced with one of its neighbors, $ni\_docs_i^{local_k}$. The neighbor is selected randomly from the documents that are very near to the current document. The selected neighbor is again subject to be replaced by another depending on a real random number within [0,1]. Thus, the selected neighbor can be altered by a new selection but now it is done from its neighbors within G^{ε}. In the onlooker phase, a document has chance to change depending on Eq. (4) and is replaced with another document selected randomly from ni_docs^{global}. In the scout phase, the abandoned documents are replaced with other documents

selected randomly in the integer interval from 1 to the total collections size. After the iterations reach to the desired limit, we gain the best document $Gdoc^{local_k}$ for each server S_k and also synonyms for each query word with variable fitnesses. The final steps of the algorithm are to determine the best synonyms for query expansion process and then to obtain the ranked lists and merge them into a single ranked list , this illustrated in the following simple algorithm.

Algorithm 3: Expand Query, Matching, and Merging Algorithm

Input: $Gdoc^{local_k}$,$S_q^{Synonym_{best}}$.

Output: Ranked list consist of the best relevant documents organized in descending order.

Expanding, Matching, and Merging Process:

Begin

Step1: Add the synonyms in $S_q^{Synonym_{best}}$ to query, name the result as $Q^{expanded}$.

Step2: Add $Gdoc^{local_k}$ to its $neighbors$ list, name the result as $docs_k^{candidate}$, $(1 \le k \le m)$.

Step3: Calculate the cosine similarity between $Q^{expanded}$ and $docs_k^{candidate}$ to produce a ranked list for each server S_k.

Step4: Merge all the ranked lists using the RR merging method.

End.

The best synonyms are determined simply by taking the largest synonyms scores. After we select the best synonyms, we expand the query by adding these synonyms to the query. Each ranked list is constructed using the similarity between the expanded query and the documents from a list consists of $Gdoc^{local_k}$ and its neighbors, $ni\_docs^{local_k}$. The merging process is achieved using the RR merging method.

VII. EXPERIMENTAL RESULTS

The proposed system is experimented on a well-known corpus called NPL used in many research works for evaluating IR systems. NPL corpus consists of 11429 documents on electronic engineering. It includes 93 queries with their own relevance judgments. University of Glasgow archives this datasets with others in a web page in [24]. The proposed system is run on a personal computer (Core-i5 @2.50 GHz, RAM 6GB, 64-bit operating system). For applying the distribution concept, NPL document collection is shuffled and divided into four parts and each one is located on a distinct server. Each sub-collection has a different size and the range for each one on a server S_k is as follows:

S_1:[1..2250], S_2:[2251..4750], S_3:[4751..7750], and S_4:[7751..11429].

The experimental tests focused on the comparison between PDEABC system and the traditional system. The given query is transformed into query vectors, each one corresponding to a local Vector Space Model (VAM) located on a remote server. The traditional system selects a subset of servers that have most probable relevant documents, searches the local inverted indexes in the corresponding servers, and then achieve the query-document similarity to find the corresponding ranked lists. After that, it achieves the results merging for getting a unified result. The selection and merging processes are performed using CORI methods. PDEABC system searches in a pseudo-random manner for the documents on the servers locally in a specific server or globally in all the servers according to PDEABC system structure. To highlight the comparison between the proposed search and the traditional search, a sample of the first ten queries is selected from NPL corpus to evaluate the results. Among the first ten queries, the traditional system fails to retrieve any relevant document with respect to some queries numbered 2,5,8,9, and 10 if the searching is conducted on two servers and it fails to respond to the queries numbered 2,5and 8 if the searching was in three servers. The PDEABC system succeeds in all queries except a one numbered 5. For illustration, Table (1) shows in detail the outcomes of both the proposed and traditional systems with respect to a query sample numbered 2.

Table 1: Performance evaluation of the NPL corpus for a query sample

System	No. of visited docs out of 11429	Ranked list (1'st ten docs in descending order)	Relevant documents (human Judgment)	Intersection	Precisions, Recalls @10	Latency (in sec)
Trad. (2 servers)	1869	[8997, 4547, 5630, 2284, 6958, 4832, 10081, 987, 7803, 5486]	[414, 1894, 3785, 4720, 5894, 6736, 7113, 7555, 7749, 7808, 8241, 8383, 9112, 9835, 10802]	[]	0.0, 0.0	0.358799
Trad.(3 Servers)	2546	Similar	Same Judgment	Similar	Similar	0.469354
PDEABC	821	[3781, 7998, 4547, 5630, 4747, 7113, 8803, 6958, 5662, 5750]	same Judgment	[7113]	0.1 , 0.066666	0.220505

The traditional system is tested once on two selected servers and again on three selected servers out of four servers. The same table can be repeated for the other nine queries. To avoid lengthiness, Table 2 shows the average evaluations of the performance with respect to all queries in the terms of the number of documents which are visited, latency, precision, recall, and the MAP measure.

Table 2: Average performance evaluation of the NPL corpus for the first ten queries

Average performance for NPL corpus	Traditional System (2 selected servers)	Traditional System (3 selected servers)	PDEABC System
Average of documents that are visited for each query out of (11429) documents.	1360	1860	754
Average of latency (Sec. /Query).	0.265112	0.351997	0.218035
Average of precision.	0.140000	0.170000	0.210000
Average of recall.	0.074223	0.134648	0.239608
MAP measure	0.271215	0.302222	0.362654

Figure (1) shows the 11-point interpolated recall-precision curves for the traditional and proposed systems for the first ten NPL queries. The curves show the emergence sequence of the documents to the user where the points at the far left are better than others.

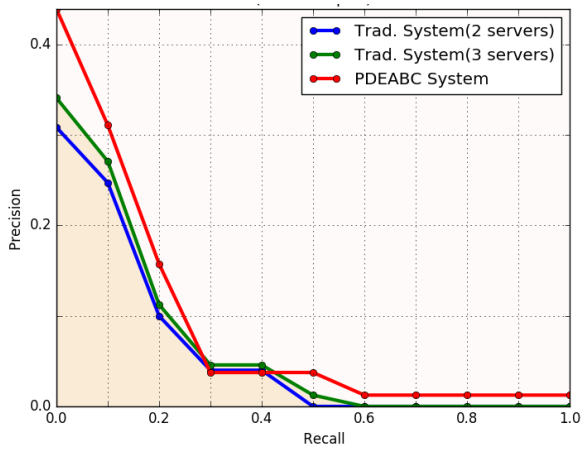

Fig.1. Average recall-precision curves for the first ten NPL queries

Mean Average Precision (MAP) combines the recall and precision into a single score, it computes the precision in a gradually manner depending on the location of relevant documents in the ranked list. i.e., it sees the first relevant document is a more important than the second in the ranked list and so on. Depending on MAP measure entered in Table 2, the PDEABC system outperforms the traditional system which has no WSD technique in a rate of 9% with reducing of latency in an approximate rate of 0.047 second for each query if the search in the traditional system is limited to half the given servers and it outperforms in a rate of 6% with reducing of latency in an approximate rate of 0.134 second for each query if the search is limited to three-quarters of the given servers.

To compute the latency, we tend to draw a simple simulation to a client-server database structures. The traditional system should be sacrificed at least one server for reduce the response time; this is at the expense of accuracy certainly. PDEABC system try all servers but in an incomplete manner using a metahuristc search to gain a good accuracy to some extent while retaining a time of relatively low latency. Since the meta-heuristic search give near optimal solution due to the random search nature, we can see some drops in the curve of the proposed system. However, the most of the times it achieves results have superiority on traditional system.

VIII. CONCLUSIONS

In this paper, we developed a mechanism to address the resource selection and the results merging which are the two major problems in the distributed information retrieval (DIR) systems. PDEABC system is proposed to overcome these two problems. PDEABC system improves the DIR efficiency through using a meta-heuristic algorithm called Artificial Bee Colony (ABC) and improving the result quality through a query expansion. The neighbor nodes are constructed using the nearest neighbor graph that improves ABC performance. For expanded the query, we used a proposed mechanism to disambiguate the ambiguous query words. After the query words are disambiguated, the synonyms are automatically selected depending on another proposed mechanism. The experimental evaluations demonstrate the superiority of the proposed system on the traditional system.

REFERENCES

[1] Gupta,Y., Saini, A., & Saxena, A.K.(2015). A new fuzzy logic based ranking function for efficient Information Retrieval system. *Expert Systems with Applications*, 42(3), 1223-1234.

[2] Sara, B. & Larbi, G. (2016). Selection of Relevant Servers in Distributed Information Retrieval System. *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, 10(5), 724-728.

[3] Rasolofo, Y., Abbaci, F., & Savoy, J. (2001). Approaches to collection selection and results merging for distributed information retrieval. *In Proceedings of the tenth international conference on Information and knowledge management*, ACM Press, 191-198.

[4] Callan, J. (2000). Distributed information retrieval. *In W. B. Croft, editor, Advances in Information Retrieval. Kluwer Academic Publishers*, (Chapter 5), 127-150.

[5] Baeza-Yates, R. & Ribiero-Neto, B. (1999). Modern Information Retrieval. *Addison Wesley Longman Publishing Co. Inc.(ACM)*, 1st ed., (Chapter 1), 1-17.

[6] Drias, H. & Mosteghanemi, H. (2010). Bees Swarm Optimization based Approach for Web Information Retrieval. *IEEE/WIC/ ACM International Conference on Web Intelligence and Intelligent Agent Technology*,1, 6-13.

[7] Drias, H. (2011). Parallel Swarm Optimization for Web Information Retrieval. *IEEE Third World Congress on Nature and Biologically Inspired Computing*, 249-254.

[8] Abdul-Hassan, A.K. & Hadi, M.J. (2016).Sense-Based Information Retrieval Using Artificial Bee Colony Approach. *International Journal of Applied Engineering Research*, 11(15), 8708-8713.

[9] Abdul-Hassan, A.K. & Hadi, M.J. (2017). Sense-Based Information Retrieval Using Fuzzy Logic and Swarm Intelligence. International Journal of Multimedia and Ubiquitous Engineering 12(1), 363-376.

[10] Cacheda, F., Carneiro, V., Plachouras, V., & Ounis, I. (2007). Performance analysis of distributed information retrieval architectures using an improved network simulation model. *Information Processing and Management*, 43(1), 204–224.

[11] Ghansah, B. & Wu, S. (2015). Survey On Score Normalization : A Case Of Result Merging In Distributed Information Retrieval. *WSEAS Transactions on Information Science and Applications*, 12, 138-147.

[12] Paltoglou, G. (2009). Algorithms and strategies for sources selection and results merging (collection fusion algorithms) in distributed information retrieval systems. *Phd thesis, Department of Applied Informatics, University of Macedonia, Thessaloniki*, (Chapter 2), 40-80.

[13] Karaboga D. (2005). An idea based on honey bee swarm for numerical optimization. Technical Report TR06, *Computer Engineering, Department, Erciyes University, Turkey.*

[14] Zhang, C., Ouyang, D., & Ning, J. (2010). An artificial bee colony approach for clustering. Expert Systems with Applications, 37(7), 4761–4767.

[15] Tankasala, G.R.(2012). Artificial Bee Colony Optimization for Economic Load Dispatch of a Modern Power system. *International Journal of Scientific & Engineering Research*, 3(1), 1-6.

[16] Koutrika, G., Liu, L., & Simske, S. (2015).Generating reading orders over document collections. *IEEE 31st International Conference on Data Engineering*, 507-518.

[17] Callan, J. P., Lu, Z., & Croft, W. B. (1995). Searching Distributed Collections with Inference Networks. *Proceedings of the ACM-SIGIR*, 95, 21-28.

[18] Xu, J. & Callan, J. P. (1998). Effective Retrieval with Distributed Collections. Proceedings of the ACM-SIGIR, 98, 112-120.

[19] Shokouhi, M. & Si, L. (2011). Federated Search. *Foundations and Trends® in Information Retrieval*, 5(1), 1–102.

[20] Saoud, Z. & Kechid, S. (2016). Integrating social profile to improve the source selection and the result merging process in distributed information retrieval. *Information Sciences*, 336(C) ,115-128.

[21] Tjin-Kam-Jet, K.T.T.E. (2009). Result Merging for Efficient Distributed Information Retrieval. *Master thesis, University of Twente*,(Chapter 3),13-17.

[22] Si, L. & Callan, J.(2003).A semisupervised learning method to merge search engine results. *ACM Transactions on Information Systems*, 4(21), 457–491.

[23] Ramya, C. & Shreedhara, K.S. (2016).A Brief Review On The Application Of Swarm Intelligence To Web Information Retrieval. *International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)*, 3(1), 60-63.

[24] Sanderson, M. (2010). Test Collection Based Evaluation of Information Retrieval Systems. *Now Publishers Inc ,4(4), 247–375.*
http:/ /ir.dcs.gla.ac.uk/resources/test_collections

الخلاصة- *استرجاع المعلومات الموزعة (DIR) هو نموذج يتيح للمستخدم الوصول إلى العديد من قواعد البيانات للبحث في مواقع مختلفة. DIR هو أكثر تعقيدا من استرجاع المعلومات المركزية (IR). فهو يتطلب معالجة مشكلتين إضافيتين هامتين هما اختيار الموارد ودمج النتائج. لقد نُشرت العديد من التقنيات لمعالجة المشكلتين في الأدب. ومع ذلك، فإنها لا تزال لها تأثير سلبي على جودة الاسترجاع ووقت الاستجابة. تهدف هذه الورقة إلى تحسين كفاءة DIR من خلال استخدام خوارزمية ارشادية عليا معينة وتحسين جودة النتائج من خلال توسع الاستعلام. وقد تم تعزيز الخوارزمية باستخدام مخطط الجار الاقرب من أجل تحسين أداء البحث. أداء النظام المقترح يفوق الأداء في النظام التقليدي بمعدل من 6٪ إلى 9٪ في حين يقلل من التاخير بمعدل تقريبي من 0.047 إلى 0.134 ثانية لكل استعلام.*