Neural Computing and Applications (2023) 35:12599–12610
https://doi.org/10.1007/s00521-023-08412-4

ORIGINAL ARTICLE

Combining variable neighborhood with gradient ascent for learning to rank problem

Osman Ali Sadek Ibrahim[1] · Eman M. G. Younis[2]

Received: 9 October 2022 / Accepted: 13 February 2023 / Published online: 4 March 2023

Abstract
Metaheuristic applications for information retrieval research are limited in spite of the importance of this problem domain. Ranking the retrieved documents based on their importance is a vital issue for the scientific and industrial communities. This paper proposes a novel variable neighborhood search (VNS) algorithm with adaptation based on an objective function for the learning to rank (LTR) problem. VNS is a global optimum metaheuristic algorithm that has been engaged to evolve the optimal solutions for heuristic problems based on exploring better neighbor solutions from the current one. The changes from the current to the next optimal solution are made during the perturbation stage to identify the global optimal solutions. The exploration procedure has been made through various mutation step sizes, whereas the exploitation process has been done by checking the quality of the evolved solutions using the fitness function. This research proposes a novel version of VNS based on four random probability distributions with gradient ascent. In addition to using the traditional random generator with gradient ascent for modifying the mutated genes of the neighborhood candidate solution in the following evolving iteration. This novel method in LTR is called gradient variable neighborhood (GVN). In the experiments, we utilized Microsoft Bing search (MSLR-WEB30K), Yahoo, and TREC Million Queries Competitions in 2008 and 2007 (LETOR 4) datasets. From the findings of the results, we can deduce that the GVN method outperformed recent studies on LTR methods.

Keywords Ranking problem · Gradient variable neighborhood · Information retrieval · Microsoft Bing · Yahoo

1 Introduction

Variable neighborhood search (VNS) is a metaheuristic algorithm suggested in [1, 2]. This method is one of the local search (LS) algorithms. However, it is a global optimization algorithm method that can overcome being stuck in local optimum solutions. This can be accomplished by more search divergence of various neighbor solutions with nondeterministic and variable mutation steps in the evolving procedure.

✉ Osman Ali Sadek Ibrahim
osman.ibrahim@mu.edu.eg

Eman M. G. Younis
eman.younas@mu.edu.eg

[1] Computer Science Department, Minia University, Minya, Egypt

[2] Information Systems Department, Minia Univeristy, Minya, Egypt

To find an optimum solution, metaheuristic algorithms have been used for improving the suggested solution. The improvement is based on checking the quality of the solution with the objective function in each iteration. This is done until no further improvements can be achieved. The VNS algorithm utilizes a strategy based on dynamically shifting neighborhood boundaries. There are numerous levels of freedom in designing variations and specific extensions of this algorithm. An enhanced solution s' in the neighborhood N(s) of the current candidate solution s can be gained at each learning iteration by irregular mutation step sizes. This causes the evolved solutions to jump and avoid the stuck in local optimal to globally optimal solutions.

The reason for that is doing more exploration for search solution space which can be achieved by these nondeterministic mutation step sizes. This VNS is a modified version of LS methods that are stuck in the local optimal solution. In addition, several metaheuristics algorithms have been suggested in recent years that extend this approach in many ways to avoid them from being stuck in local maxima or minima solutions. Examples of these methodologies which were investigated in previous research studies for other research domains are: evolutionary strategies [3], evolutionary programming [4], genetic algorithms [5], and evolutionary algorithms [6]. However, VNS showed the best performance among these approaches in other problem domains [7, 8].

Intensive experimental studies have been conducted in previous research using metaheuristic methods for obtaining optimal evolved solutions. However, limited research studies of these categories have been developed for learning to rank (LTR) research in information retrieval (IR) [9–11]. This study introduces a novel method for the VNS algorithm which can accomplish enhancement in the accuracy of ranking documents in IR. This paper used two evaluation fitness metrics to measure the accuracy of the IR system which are the mean average precision (MAP) metric and the normalized discounted cumulative gain for the top-10 WebPages retrieved (NDCG@10) metric. The suggested approach for adapting the candidate solution, by determining a better neighborhood solution to the current one, has not been used before in the computational intelligence problem domains. Furthermore, the VNS algorithm has not been applied previously in the ranking problem.

The contributions of this paper are:

1. This paper proposes a novel technique called the gradient variable neighborhood search (GVN) strategy in IR optimization research. This is achieved by utilizing 5 probability distributions which act as random number generators for mutating the neighborhood evolved solutions. It also combines gradient ascent with VNS in choosing these mutation step sizes.
2. The suggested approach is used for the first time for the LTR problem.
3. Making a comparative research study of the suggested approach and the most recent studies in the area of LTR. This is carried out by implementing the suggested approach using large and the latest known LTR datasets.
4. For reproducible research, the Java code for the suggested algorithm is provided.

This paper is organized as follows. Section 2 illustrates the background and various information retrieval (IR) problem domains. Section 3 presents related work in the literature of LTR for IR research existing in previous studies. Section 4 gives explanations of the suggested VNS approaches. whereas Sect. 5 demonstrates the results derived using Yahoo, MSLR-WEB30K, and LETOR 4 (TREC Million Query 2008 and TREC Million Query 2007) datasets. Section 6 concludes the paper's results and suggests future research avenues.

2 Background

This section illustrates different research problems of IR systems in which researchers used computational intelligence (CI) methods. These methods have been used to optimize them for better IR performance. First, it introduces a high-level overview of various IR system problem domains. Then, it concentrates on the ranking problem domain in which learning to rank applications was used in this study.

2.1 Information retrieval research problem domains

Figure 1 illustrates the main research problem domains existing in IR and search engines. Artificial intelligence (AI) techniques were applied in previous research. In this subsection, brief definitions for these problems concentrating on the LTR problem. More details for other various IR problem domains in which computational intelligence methods were used to optimize them are existing in [12].

From Fig. 1, the main problem domains in which AI methods were used are highlighted by dotted shapes. These domains are document or WebPage representation, user query representation, user profile feedback, and ranking the retrieved documents or WebPages list for users. Typically, every document or WebPage and user query is subjected to lexical analysis for extracting heuristic information content for the document or the query. These procedures are approximate methods that may result in inaccurate representations for them. Thus, AI techniques were used based on relevance labels provided by user interaction with the IR system in user query logs to optimize them. However, some documents may not have relevance labels for using AI methods to optimize document or query representation on the IR document collections [13]. This issue inspires the need of using machine learning methods for producing learning models that can be used on unseen data. This was the starting motivation point for learning to rank research for optimizing IR systems through a learned ranking model rather than optimizing query and document representation.

The learning to rank (LTR) problem is colored yellow in Fig. 1 to indicate it to researchers in which IR component exists. The following subsection has more explanation for LTR in IR systems that may inspire computer researchers to recognize this hot topic for its business value on the web.

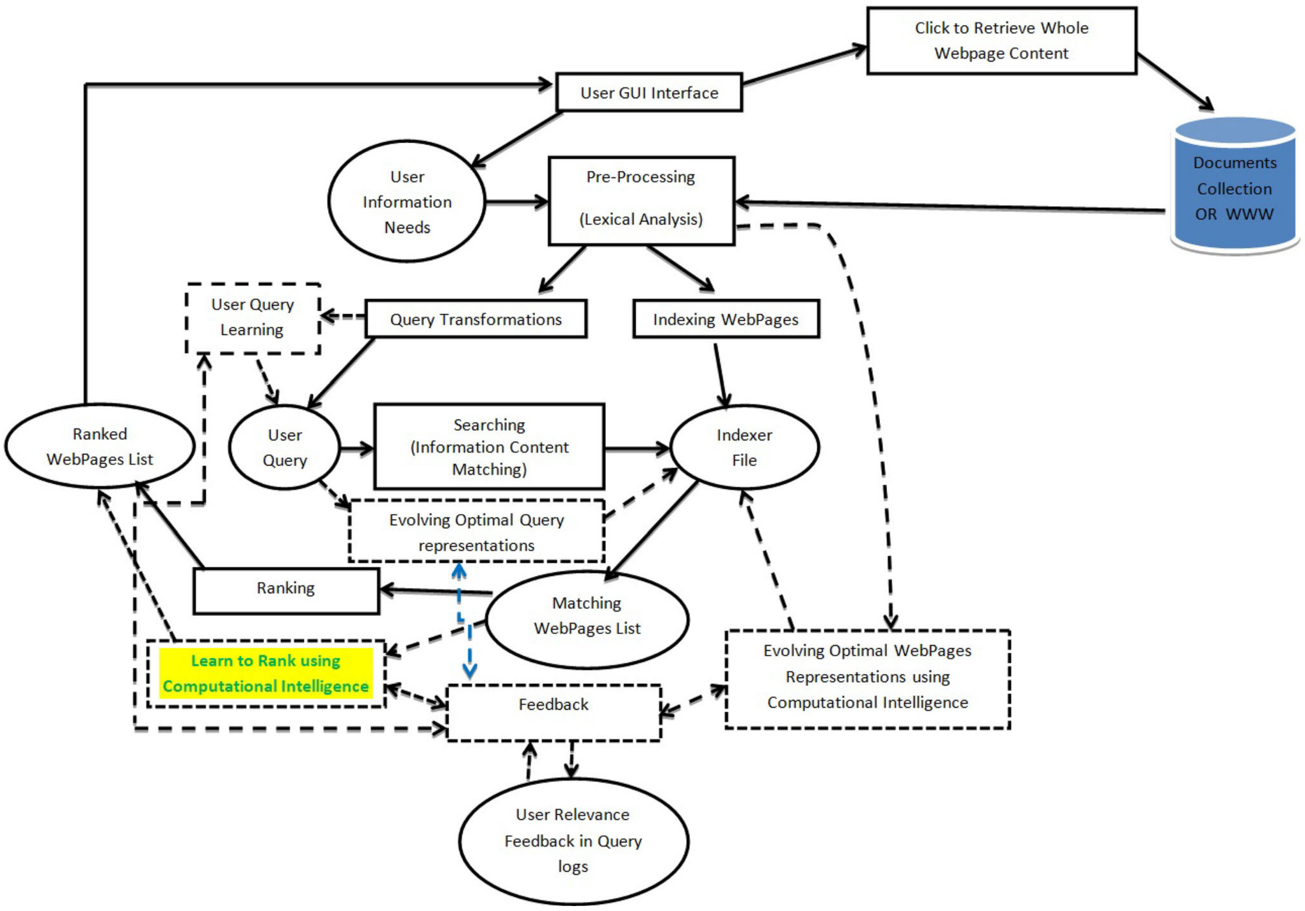
Click to Retrieve Whole Webpage Content
User GUI Interface
Documents Collection OR WWW
User Information Needs
Pre-Processing (Lexical Analysis)
User Query Learning
Query Transformations
Indexing WebPages
Ranked WebPages List
User Query
Searching (Information Content Matching)
Indexer File
Evolving Optimal Query representations
Ranking
Matching WebPages List
Learn to Rank using Computational Intelligence
Feedback
Evolving Optimal WebPages Representations using Computational Intelligence
User Relevance Feedback in Query logs

Fig. 1 Main problem domains existing in IR systems and search engines modified from figure in [13]

2.2 Learning to rank framework

The main challenge in information retrieval (IR) is ranking the retrieved documents or WebPages that respond to the user's query. They are ranked in terms of their relevance or business importance matching. Unsupervised scoring approaches such as term frequency-inverse document frequency (TF-IDF), OKAPI-BM25, and language models, among others, were used in early information retrieval studies. These methods were applied to evaluate the similarity of the retrieved documents with the queries. The limitation of using only one scoring function in IR systems was the reason for poor IR performance and reliability problems. When applying only one scoring technique in IR systems, two problems were uncovered. These issues are the approximation of using probabilistic, and language models without considering the relevance feedback, and business importance of the WebPages or documents. These motivated the usage of several scoring methods for ranking documents in accordance with the users' queries. Moreover, the business value of the documents on the web and the server hosting them, as well as other factors, should be considered when ranking the pages in terms of their usefulness.

Tao Qin et al. [14] outlined a new trend for developing LETOR datasets derived from search engines and well-known TREC conference collections. These benchmarks contain features that represent more than one weighting scheme as the WebPage information content. They also included certain additional elements, such as PageRank, which is used by Google, and the importance of the host server, which is used by search engines. Furthermore, in these new benchmarks, the pages in the datasets were mapped into rated query-document pairs. As a result, the study has moved to apply computational intelligence applications to learn optimal ranking functions. Machine learning and evolutionary computational algorithms have made extensive use of learning to rank datasets in the previous studies as in [10]. Figure 2 shows a framework for the LTR problem domain.

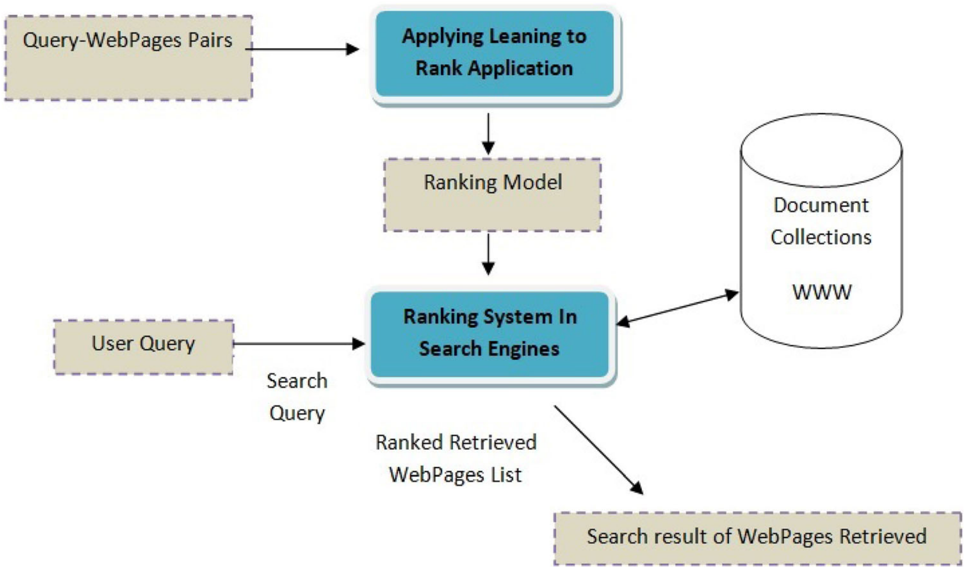
Query-WebPages Pairs
Applying Leaning to Rank Application
Ranking Model
Document Collections
WWW
User Query
Ranking System In Search Engines
Search Query
Ranked Retrieved WebPages List
Search result of WebPages Retrieved

Fig. 2 Learning to rank architecture

3 Related work

There are three main categories of LTR approaches [15–17]. The first one is called the pointwise approach. The second category is the pairwise approach. The third category is the listwise approach. This classification methodology is based on how the evaluation procedure is performed. It can be done by using either the objective function or using the loss value function. For the pointwise method, every query document is considered an item of the learning process. Gradient boosted regression trees (GBRT or MART) [18, 19], boosting [20], gradient boosted regression trees (GBRT or MART) [18, 19], linear regression (LR) [21], and random forest (RF) [22] are all considered representative examples of the pointwise approach.

On the other hand, the pairwise approach uses a set of two query-document sets for the same query as the learning item and uses a loss or objective metric function for evaluations. Examples of the pairwise approach are RankBoost, RankNET (rank neural net) [23], and support vector machine (SVMRank) [17]. On the contrary, the learning item in the listwise approaches is represented as a list of query-document sets for every query of the whole list of the search. Examples of the algorithms of the listwise approach are Coordinate Ascent [24], ListNET (Listwise Neural Net) [16], AdaRank [25], and RankGPES [26, 27].

The following methods have been suggested after these previous studies for the process of modeling user clicks [28–30]. Recently, a new research trend in LTR is called Online LTR. This new research trend studies the use of implicit relevance labels to mimic the selections of the user. Then, measure how well the online LTR method performs. This is because the actual user click selections may undergo several bias problems which were stated in [28, 29].

Recently, a lot of research has contributed to proving that learning from explicit user relevance labels performs better than learning from the implicit relevance feedback in online LTR. This was done using user simulation Click models [10, 31]. From another point of view, user click bias corrections have been studied in order to gain better performance on ranking search engine results with an improved link between dataset features and their relevance feedback labels [32, 33]. For various aspects of bias corrections, [33] presented two well-known LTR approaches from the literature. These approaches are Deep Neural Network and λ*Mart* which outperformed previous research.

In this paper, we propose a novel algorithm for merging VNS with the gradient ascent procedure in the continuous optimization research field in AI. This is accomplished using five probability distributions acting as number generators for mutating step sizes of neighboring searches for generating optimal solutions. Moreover, applying the VNS algorithm for the first time in the LTR problem by the suggested approach. Further, comparing the algorithm also suggested with recent studies in the LTR research domain [9, 10, 31–33]. This was accomplished by using two of the largest and most recent available LTR datasets from search engines and two well-established and recognized datasets from TREC document collections.

4 The suggested method

This research suggests a novel algorithm by combining the variable neighborhood search (VNS) with gradient ascent (GVN) procedure. The GVN algorithm runs in the following steps summarized below. First: a list of various neighbor offspring solutions (*OffNeighbor*) is evolved, and this novel GVN algorithm are provided in the following link GVN for enabling the research to be reproducible.

Algorithm 1: GVN-Rank: Gradient Variable Neighbourhood Search for LTR

```
Input  : training dataset $\Psi(q,d)$ composed of query and related document sets of feature
         vectors.
Output: evolved LTR model $LR(q,d)$ that allocates feature weights to each
         query-document set which measure the feature importance to its relevance level.
1  Initialization
2  for ($Gen_l \in S$) do
3      $Gen_l = 0.5$ ;
4  end
5  for $j = 1$ to $MaxGenerations$ do
6      Select the number of genes to mutate $R_{ij}$ at random from 1 to $M$ in S;
7      To evolve Neighbourhood rank model $Neighbour_j$ according to the random probability
        distribution $PD_i$ in equation 1.;
8      if (Fitness($S$,$\Psi(q,d)$) <Fitness($Neighbour_j$,$\Psi(q,d)$)) then
9          $S = Neighbour_j$;
10     else
11         $Neighbour_j = S$;
12     end
13 end
14 return evolved LTR model $LR(q,d) = S^T \bullet \Psi(q,d) = W^T \bullet \Psi(q,d)$, which is $S$ at the end
    of the $MaxGenerations$ encloses the learning vector $W_j$ of $M$ feature importance, $T$
    specifies the transpose
```

each neighbor solution of them is then evaluated one by one.

Let $Neighbour = OffNeighbour_1$, $OffNeighbour_2, \ldots, Off\ Neighbour_n$ be a *jth* generation of offspring neighbor ranking models specifying as the neighbor evolved solutions to the current candidate ranking model chromosome. The procedure in which they will be evaluated for their ranking accuracy compared to the best current solution for the current evolving iteration is called objective metric evaluation. In this fitness evaluation process, two prominent objective measures were applied which are mean average precision (MAP) and normalized discounted cumulative gain at the top 10 documents retrieved (NDCG@10). The GVN algorithm iteratively checks the quality of the neighborhood ranking solutions specified in each learning iteration, while the evolving iteration threshold operator is given by $1 \leq j \leq MaxGenerations$.

Beginning from a given candidate solution S and while *MaxGenerations* is the maximum number of evolving or learning iterations not reached. The GVN algorithm recommences search in the current neighbor chromosomes. This repeatedly happens in accordance with replacing the current candidate solution with the neighbor solution evolved if its performance is lower than the evolved solution. This happens in each evolving iteration until the stopping criterion is reached. The java archive packages for

Algorithm 1 demonstrated a high-level overview of the GVN's algorithm. This GVN approach generates the neighbor chromosome by mutating the candidate ranking model by one of five probability distributions to act as random number generators. These probability distributions are Gaussian, Cauchy, Levy, Uniform, and traditional random number methods. This has been applied as variable mutation step sizes. For each neighbor solution, the mutation step size uses a collection of random numbers from only one probability distribution for each neighbor solution. These distributions have been applied in some research studies for solving other optimization problems [3, 5, 9, 34]. Figure 3 illustrates the detailed flowchart of the GVN algorithm for more clarification.

The key terms in this flowchart in Fig. 3 are as follows:

1. i is the number of neighborhood solutions in each evolving iteration. Each neighbor solution is produced by mutation stepsize for its genes from one probability distribution by equation 1. In this paper, we used 5 probability distributions. Thus,i = 5 in each evolving iteration.
2. PD_i represents the probability distribution i as a random generator number. This paper uses Gaussian, Cauchy, Levy, Uniform, and traditional random

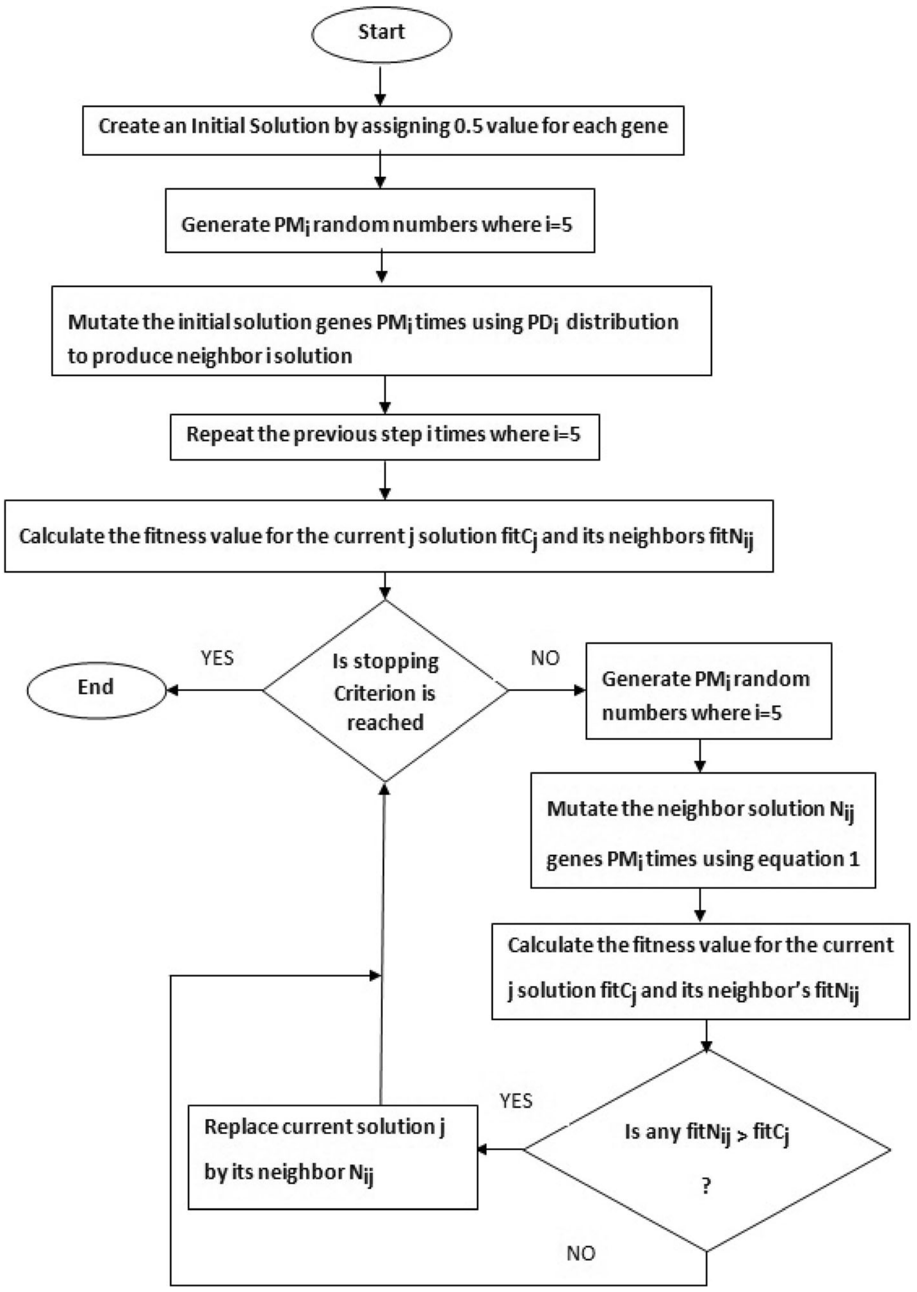
Start
Create an Initial Solution by assigning 0.5 value for each gene
Generate PMi random numbers where i=5
Mutate the initial solution genes PMi times using PDi distribution to produce neighbor i solution
Repeat the previous step i times where i=5
Calculate the fitness value for the current j solution fitCj and its neighbors fitNij
Is stopping Criterion is reached
YES
End
NO
Generate PMi random numbers where i=5
Mutate the neighbor solution Nij genes PMi times using equation 1
Calculate the fitness value for the current j solution fitCj and its neighbor's fitNij
Is any fitNij > fitCj ?
YES
Replace current solution j by its neighbor Nij
NO

Fig. 3 Gradient variable neighborhood flowchart for learning to rank problem

number generators as a random generators for determining mutation step size.

3. PM_i is a random number that represents the probability of mutations in neighbor solution i.
4. C_j is the best current evolved ranking model for evolving iteration j, while N_{ij} is its neighbor solution produced by PD_i in the same evolving iteration j.
5. $fitC_j$ and $fitN\ ij$ are the fitness evaluation values for the current solution C_j and its neighbor N_{ij}.

$$Mutated\_Gene_{ij} = Gene_{ij} + \ PD_i * (fitC_j - fitN_{ij}) \qquad (1)$$

5 Experimental results

The purpose of this section is to illustrate the experimental findings and compare the performance of the suggested GVN approach with previously presented Metaheuristic and ML techniques. The datasets used in the experiments include training, validation, and test data segments, used for building the models, evaluating, and measuring their performance. To determine the most efficient ranking model, we did the following: first, the proposed GVN LTR method is implemented for training or improving a solution. Then, the effectiveness of the evolved solution is evaluated using new data to measure its predictive power.

5.1 Datasets and performance evaluation measures

The experimental research presented here uses the LTR benchmarks: MSLR-WEB30K extracted from Bing search engine, Yahoo and LETOR 4 (MQ2007 and MQ2008) dataset from TREC document collection [14, 35–37]. The main features of these datasets are shown in Table 1. These are considered the largest available learning to rank datasets for research such as Microsoft Bing search datasets (MSLR-WEB30K) and Yahoo. There are many attributes included such as low-level features like cumulative term frequency (CTF) and cumulative inverse document frequency (CIDF) of the document terms existing in each combination of query and the corresponding document(s). For every part of the document, the basic features were calculated for the document title, anchor, body, and for the rest of the page text.

Features common between the query and corresponding document are called high-level features. These features compute the similarity between the query and the documents. There are also some features related to language models such as absolute discounted smoothing (LMIR.-ABS), Jelinek-Mercer smoothing (LMIR.JM), Bayesian smoothing using Dirichlet priors (LMIR.DIR), and click user interaction-related features which have been demonstrated in previous IR studies [14, 35–37] as features.

Each query has a number of documents that can be either relevant or irrelevant, which constituted so-called query-document sets or pairs. For each query-document set or pair, the relevance label measures the user preference levels for the query to that document. The values of relevance labels usually range between zero and two. The value of zero refers to irrelevant, whereas one refers to partially relevant and two refers to completely irrelevant between the query and the corresponding document. The MSLR-WEB30K Benchmark, collected from the Bing search engine, and Yahoo search dataset have relevance labels ranging from zero (irrelevant) to four (perfectly relevant), while LETOR 4 Benchmark have only three levels of relevance ranging from irrelevance to totally relevance.

This research utilizes two objective performance evaluation measures to evaluate the results in this paper. These measures are mean average precision (MAP) and normalized cumulative gain at the top-10 documents retrieved (NDCG@10). Both fitness measures have been used extensively in previous research studies such as in [11, 32, 33, 36]. The MAP metric only determines whether the query document retrieved is relevant. But, it does not evaluate the levels of graded relevance for each retrieved document. Instead of only focusing on just the top-k query-document pairs, MAP evaluates the average precision levels over the complete set of search list obtained results. The graded relevance degree for each pair of the retrieved top-k query documents is considered by the NDCG@K metric.

In the next subsection, we provide a comprehensive analysis of the results in this research. The analysis is composed of two parts for both research communities' point of views. First, from the metaheuristic researchers' point of view and from ML researchers' point of view. Thus, the paper includes an evaluation of the presented GVN capabilities as a metaheuristic method in evolving optimal ranking models. In addition to evaluating GVN when applied to the training and validating dataset besides evaluating their performance for obtaining the optimal predictive results on the unseen test data.

In this research, we compare the results obtained from the novel proposed GVN with the recent state-of-the-art previous research from the literature which outperformed other available methods [10, 31]. The first method in the comparison is (1+1)-evolutionary strategy LTR (ESRank), which outperformed fourteen computational intelligence methods as demonstrated in [31].

Whereas extending it by combining it with simulated annealing (SASRank) or gradient mutation step size added (EGSRank) have been suggested in recent studies for better efficiency [9, 10]. Moreover, recent research suggested correcting the bias of the user relevance feedback for better performance on the best known *λMart* and deep neural network (DNN) technique among approaches which are provided in [32, 33]. These techniques with their variants were provided in the comparative study with the GVN algorithm presented in this paper.

Table 1 Characteristics of the LTR datasets used in this work

Dataset name	Queries	No. of data items	Features	Relevance labels	No. of folds
MQ2007	1692	69,623	46	{0, 1, 2}	5
MQ2008	784	15,211	46	{0, 1, 2}	5
Yahoo S1	29,921	709,877	519	{0, 1, 2, 3, 4}	1
MSLR-WEB30K	30,000	3,771,125	136	{0, 1, 2, 3, 4}	5

Table 2 Average of the training performance for metaheuristic algorithms implemented on 4 datasets with the MAP evaluation measure

The MAP results gained by training data					
Algorithms	Yahoo S1	MSLR-WEB30K	MQ2008	MQ2007	Winner numbers
ESRank with Gaussian	0.841	0.576	0.472	0.449	0
ESRank with Cauchy	0.833	0.493	0.445	0.432	0
ESRank with Levy	0.840	0.582	0.477	0.453	0
ESRank with Uniform	0.830	0.492	0.457	0.421	0
SASRank with Gaussian	0.833	0.581	0.481	**0.456**	1
SASRank with Cauchy	0.834	0.568	0.479	0.45	0
SASRank with Levy	0.828	0.578	0.449	0.452	0
SASRank with Uniform	0.838	**0.593**	0.476	0.453	1
EGSRank with Gaussian	0.837	0.524	0.457	0.448	0
EGSRank with Cauchy	0.830	0.526	0.47	0.449	0
EGSRank with Levy	0.827	0.522	0.456	0.435	0
EGSRank with Uniform	0.835	0.543	0.471	0.433	0
GVN	**0.86**	0.586	**0.502**	0.454	2

Bold values indicate the highest effectiveness in the corresponding fitness evaluation metric

Table 3 Average of validation performance of metaheuristic algorithms implemented on 4 datasets with the MAP evaluation measure

The MAP results gained by validation data					
Algorithms	Yahoo S1	MSLR-WEB30K	MQ2008	MQ2007	Winner numbers
ESRank with Gaussian	0.835	0.579	0.530	0.48	0
ESRank with Cauchy	0.827	0.528	0.492	0.457	0
ESRank with Levy	0.835	0.583	**0.547**	0.482	1
ESRank with Uniform	0.824	0.507	0.508	0.444	0
SASRank with Gaussian	0.827	0.593	0.52	0.473	1
SASRank with Cauchy	0.827	0.586	0.519	0.471	0
SASRank with Levy	0.834	0.576	0.484	0.47	0
SASRank with Uniform	0.831	0.598	0.503	0.466	0
EGSRank with Gaussian	0.832	0.546	0.522	0.479	0
EGSRank with Cauchy	0.825	0.536	0.527	0.465	0
EGSRank with Levy	0.820	0.554	0.508	0.459	0
EGSRank with Uniform	0.828	0.542	0.508	0.46	0
GVN	**0.846**	**0.634**	0.544	**0.501**	3

Bold values indicate the highest effectiveness in the corresponding fitness evaluation metric

5.2 Main findings and discussion

The purpose of this subsection is to analyze the results obtained from the experiments and to compare the GVN approach performance with other metaheuristic algorithms to prove its effectiveness. The results of the evolving procedure are obtained from applying the suggested GVN method to the training and validation data. Whereas the predictive results are applied to the previously unseen Benchmark test data. Many comparative studies among various metaheuristic techniques have been done in other optimization problem domains. In these studies, researchers checked the quality of the evolved optimal or near-optimal solutions only on the training data. In this work experiments, rather than testing only on training, we evaluated the proposed methods using training, validation, and test data.

Thus, this paper includes the evolving results implemented on the training and validating data. Moreover, the results of the predictive evaluation on test data are also presented. The results of the training, validation, and testing are presented in Tables 2, 3, 4, 5, 6 and 7. Then, the paper compares the suggested method GVN performance with the most recent and best-performing approaches from previous studies [32, 33]. Table 8 shows the comparative results for comparing this work with state-of-the-art methods.

Table 4 Average of predictive efficiency of metaheuristic algorithms implemented on 4 datasets using MAP evaluation measure

The MAP results gained by test data

Algorithms	Yahoo S1	MSLR-WEB30K	MQ2008	MQ2007	Winner numbers
ESRank with Gaussian	0.845	0.589	0.452	**0.487**	1
ESRank with Cauchy	0.837	0.527	0.435	0.465	0
ESRank with Levy	0.844	0.592	0.461	0.486	0
ESRank with Uniform	0.834	0.496	0.447	0.447	0
SASRank with Gaussian	0.838	0.583	0.453	0.485	0
SASRank with Cauchy	0.837	0.595	0.463	0.476	0
SASRank with Levy	0.838	0.589	0.451	0.485	0
SASRank with Uniform	0.842	0.598	0.458	0.480	0
EGSRank with Gaussian	0.84	0.549	0.452	0.482	0
EGSRank with Cauchy	0.834	0.527	0.46	0.479	0
EGSRank with Levy	0.831	0.549	0.450	0.466	0
EGSRank with Uniform	0.836	0.533	0.462	0.462	0
GVN	**0.857**	**0.609**	**0.486**	0.485	3

Bold values indicate the highest effectiveness in the corresponding fitness evaluation metric

Table 5 Average of training efficiency of metaheuristic algorithms implemented on 4 datasets with NDCG@10 evaluation measure

The NDCG@10 results gained by training data

Algorithms	Yahoo S1	MSLR-WEB30K	MQ2008	MQ2007	Winner numbers
ESRank with Gaussian	0.705	0.387	0.499	0.428	0
ESRank with Cauchy	0.676	0.264	0.481	0.413	0
ESRank with Levy	0.706	0.394	0.506	0.428	0
ESRank with Uniform	0.674	0.256	0.486	0.403	0
SASRank with Gaussian	0.770	0.448	**0.559**	**0.489**	2
SASRank with Cauchy	0.755	0.472	0.548	0.488	0
SASRank with Levy	0.765	0.398	0.526	0.428	0
SASRank with Uniform	0.771	0.495	0.552	0.488	0
EGSRank with Gaussian	0.749	0.437	0.550	0.467	0
EGSRank with Cauchy	0.745	0.365	0.544	0.481	0
EGSRank with Levy	0.754	0.422	0.534	0.466	0
EGSRank with Uniform	0.744	0.353	0.544	0.479	0
GVN	**0.786**	**0.497**	0.549	0.488	2

Bold values indicate the highest effectiveness in the corresponding fitness evaluation metric

The highlighted bold results in tables from 2, 3, 4, 5, 6 and 7 show the best-evaluated values obtained on the benchmarks. The last column of these tables contains the number of Winner Numbers for the best performances gained by each method. Generally, the GVN approach performed better than the other techniques. The winner number of best performances using GVN is 16 (8 MAP and 8 NDCG@10) out of 24 evaluation fitness values. While the second approach that obtains the best performance is SASRank with its mutation variations by 6 (3 MAP and 3 NDCG@10) values. The reason for the effective results obtained by GVN is that there is more exploration by various probability distributions used as random generators. They were included in each neighbor evolved solution for the mutation process in GVN. Thus, GVN can jump from a locally optimal solution to the globally optimal one.

In Table 8, a comparative results between recent research studies in ranking problem domain with the proposed GVN method [32, 33]. The techniques utilized in this comparison are online and offline approaches. Dueling bandit gradient descent (DBGD), multileave gradient descent (MGD), and pairwise differentiable gradient descent (PDGD) are online LTR techniques used in this comparison. The offline approaches used in this comparison are two well-known techniques with their correction of bias methods. These approaches are *λMart*, *λMart* with affine correction (AC), *λMart* with mixture-based correction (MBC), deep neural networks (DNNs), DNN with AC, and DNN with MBC. The details of these methods are

Table 6 Average of validation efficiency of metaheuristic algorithms implemented on 4 datasets with NDCG@10 evaluation measure

The NDCG@10 results gained by validation data					
Algorithms	Yahoo S1	MSLR-WEB30K	MQ2008	MQ2007	Winner numbers
ESRank with Gaussian	0.703	0.398	0.563	0.458	0
ESRank with Cauchy	0.672	0.279	0.541	0.437	0
ESRank with Levy	0.703	0.41	0.508	0.459	0
ESRank with Uniform	0.671	0.259	0.536	0.437	0
SASRank with Gaussian	0.763	0.489	0.583	0.508	0
SASRank with Cauchy	0.754	0.476	0.609	0.506	0
SASRank with Levy	0.763	0.437	0.565	0.461	0
SASRank with Uniform	0.766	0.473	0.601	**0.520**	1
EGSRank with Gaussian	0.744	0.424	0.608	0.485	0
EGSRank with Cauchy	0.742	0.363	0.610	0.504	0
EGSRank with Levy	0.75	0.414	0.604	0.49	0
EGSRank with Uniform	0.742	0.356	0.602	0.505	0
GVN	**0.769**	**0.498**	**0.617**	0.517	3

Bold values indicate the highest effectiveness in the corresponding fitness evaluation metric

Table 7 Average of the predictive efficiency of metaheuristic algorithms implemented on 4 datasets with NDCG@10 fitness measure

The NDCG@10 results gained by test data					
Algorithms	Yahoo S1	MSLR-WEB30K	MQ2008	MQ2007	Winner numbers
ESRank with Gaussian	0.707	0.361	0.478	0.462	0
ESRank with Cauchy	0.679	0.263	0.467	0.440	0
ESRank with Levy	0.708	0.394	0.479	0.453	0
ESRank with Uniform	0.677	0.259	0.466	0.436	0
SASRank with Gaussian	0.773	0.426	0.522	0.526	0
SASRank with Cauchy	0.758	0.439	0.522	0.516	0
SASRank with Levy	0.765	0.392	0.515	0.469	0
SASRank with Uniform	0.774	0.461	**0.53**	0.519	1
EGSRank with Gaussian	0.753	0.435	0.524	0.502	0
EGSRank with Cauchy	0.748	0.368	0.516	0.514	0
EGSRank with Levy	0.758	0.398	0.508	0.490	0
EGSRank with Uniform	0.748	0.370	0.522	0.508	0
GVN	**0.783**	**0.496**	0.521	**0.539**	3

Bold values indicate the highest effectiveness in the corresponding fitness evaluation metric

introduced in [32, 33]. In this table, the results gained from our technique are compared with the best NDCG@10 results gained by the researchers in previous research. From these results, we can find that the GVN outperformed the other techniques in three benchmarks (Yahoo, MSLR-WEB30K, and MQ2007), while DBGD (linear model) outperformed other approaches applied to MQ2008 dataset.

6 Conclusions and future work

To conclude, this paper presented a new variable neighborhood-based method, which is merging variable neighborhood with gradient ascent called (GVN). In the GVN approach, a specific probability distribution was used for making all mutations in the candidate solution that are different from the mutation step size for other neighborhood solutions.

In the GVN approach, more exploration of searching for maximal solutions was accomplished using various probability distributions for mutation step size in neighborhood solutions. A global maximum solution can be found using this methodology without getting stuck in local maximum solutions by requiring less exploration of the solution space.

From the results, We can also conclude that GVN outperformed other LTR approaches of recent and most known LTR methods. Results here were illustrated from

Table 8 Comparison among the suggested GVN approach and recent algorithms

The predictive NDCG@10 results obtained on test data					
Algorithms	Yahoo S1	MSLR-WEB30K	MQ2008	MQ2007	Winner numbers
GVN	**0.783**	**0.496**	0.521	**0.539**	3
DBGD (linear)	0.683	0.332	**0.694**	0.484	1
DBGD (neural)	0.677	0.319	0.671	0.464	0
MGD (linear)	0.715	0.334	0.691	0.493	0
Pairwise (linear)	0.709	0.316	0.673	0.479	0
PDGD (linear)	0.737	0.428	0.689	0.512	0
PDGD (neural)	0.734	0.431	0.689	0.509	0
$\lambda Mart$	0.692	0.403	NA	NA	0
$\lambda Mart$ AC	0.763	0.477	NA	NA	0
$\lambda Mart$ MBC	0.767	0.474	NA	NA	0
DNN	0.716	0.415	NA	NA	0
DNN AC	0.746	0.448	NA	NA	0
DNN MBC	0.744	0.440	NA	NA	0

Bold values indicate the highest effectiveness in the corresponding fitness evaluation metric

the point of view of metaheuristic research and ML research. According to these results, GVN provides 16 best fitness values (8 MAP and 8 NDCG@10) values out of 24 evaluation metric values. On the other hand, SASRank has the second best fitness values by 6 mutation variations (3 MAP and 3 NDCG@10) values, whereas ESRank approach is the third fitness performance by 2 (2 MAP) values.

In this paper, we also compared the suggested approach with recent machine learning studies. From this comparison, we can conclude that GVN outperformed the other approaches applied to the three datasets (Yahoo, MSLR-WEB30K, and MQ2007), whereas DBGD (linear model) outperformed other methods with MQ2008 dataset. The future direction of research is to browse the capabilities of other novel metaheuristic methods for achieving better performance. In addition to exploring the field of building explainable and interactive artificial intelligence techniques for IR systems.

Author contributions All authors wrote and reviewed the paper.

Funding Open access funding provided by The Science, Technology & Innovation Funding Authority (STDF) in cooperation with The Egyptian Knowledge Bank (EKB).

Data availibility statement The source code is available in section 4, while the datasets are existing on the Public Microsoft LETOR website.

Declarations

Conflict of interest The authors declare that there does not exist any kind of conflict of interest.

Ethical approval There does not exist any kind of human or animal participation in this research.

References

1. Papalitsas C, Karakostas P, Andronikos T (2019) A performance study of the impact of different perturbation methods on the efficiency of GVNS for solving TSP. Appl Syst Innov 2(4):2571–5577. https://doi.org/10.3390/asi2040031
2. Mladenović N, Hansen P (1997) Variable neighborhood search. Comput Oper Res 24(11):1097–1100. https://doi.org/10.1016/S0305-0548(97)00031-2
3. Kramer O (2016) Machine learning for evolution strategies, vol 20. Springer, New York
4. Fogel DB (1999) An overview of evolutionary programming. In: Evolutionary algorithms, Springer, pp 89–109
5. M Zbigniew (1996) Genetic algorithms+data structures = evolution programs, 3rd edn. Springer, Berlin, Heidelberg, p 3540606769
6. Nannen V, Smit SK, Eiben AE (2008) Costs and benefits of tuning parameters of evolutionary algorithms. In: International conference on parallel problem solving from nature, Springer, p 528–538
7. Montoya-Torres JR, Moreno-Camacho CA, Vélez-Gallego MC (2022) Variable neighbourhood search for job scheduling with

position-dependent deteriorating processing times. J Oper Res Soc 1–15

8. Pinheiro RL, Landa-Silva D, Atkin J (2016) A variable neighbourhood search for the workforce scheduling and routing problem. In: Pillay N, Engelbrecht AP, Abraham A, du Plessis MC, Snášel V, Muda AK, (eds) Advances in nature and biologically inspired computing p 247–259, Cham, Springer. ISBN 978-3-319-27400-3
9. Ibrahim OA (2022) Exploration and exploitation methods using (1+1)-evolutionary algorithms for better ranking models. https://doi.org/10.21203/rs.3.rs-1968676/v1
10. Ibrahim OA, Younis EMG (2022) Hybrid online–offline learning to rank using simulated annealing strategy based on dependent click model. Knowl Inf Syst 1–15
11. Ibrahim OA, Landa-Silva D (2018) An evolutionary strategy with machine learning for learning to rank in information retrieval. Soft Comput. https://doi.org/10.1007/s00500-017-2988-6
12. Ibrahim OA (2022) A survey on computational intelligence applications in information retrieval. URL https://doi.org/10.21203/rs.3.rs-2334978/v1
13. Ibrahim OA, Landa-Silva D (2016) Term frequency with average term occurrences for textual information retrieval. Soft Comput 20(8):3045–3061. https://doi.org/10.1007/s00500-015-1935-7
14. Qin T, Liu TY, Xu J, Li H (2010) Letor: a benchmark collection for research on learning to rank for information retrieval. Inf Retr 13(4):346–374. https://doi.org/10.1007/s10791-009-9123-y
15. Liu Tie-Yan (2009) Learning to rank for information retrieval. Found Trends Inf Retr 3(3):225–331
16. Cao Z, Qin T, Liu TY, Tsai MF, Li H (2007) Learning to rank: from pairwise approach to listwise approach. In: Proceedings of the 24th international conference on machine learning, ICML '07, p 129–136, New York, NY, USA. ACM. ISBN 978-1-59593-793-3. https://doi.org/10.1145/1273496.1273513
17. Li H (2014) Learning to rank for information retrieval and natural language processing, Second Edition. Morgan and Claypool Publishers. ISBN 9781627055857
18. Friedman JH (2001) Greedy function approximation: a gradient boosting machine. Ann Stat 29(5):1189–1232
19. Mohan A, Chen Z, Weinberger KQ (2011) Web-search ranking with initialized gradient boosted regression trees. In: Journal of machine learning research, workshop and conference proceedings, vol 14. pp 77–89
20. Freund Y, Iyer R, Schapire RE, Singer Y (2003) An efficient boosting algorithm for combining preferences. J Mach Learn Res 4:933–969
21. Yan X, Su XG (2009) Linear regression analysis: theory and computing. World Scientific Publishing Co., Inc, River Edge, NJ, USA
22. Leo B (2001) Random forests. Mach Learn 45(1):5–32. https://doi.org/10.1023/A:1010933404324
23. Burges C, Shaked T, Renshaw E, Lazier A, Deeds M, Hamilton N, Hullender G (2005) Learning to rank using gradient descent. In: Proceedings of the 22nd international conference on machine learning, ICML '05, pp 89–96, New York, NY, USA. ACM. ISBN 1-59593-180-5. 10.1145/1102351.1102363
24. Metzler D, Bruce Croft W (2007) Linear feature-based models for information retrieval. Inf Retr 10(3):257–274. https://doi.org/10.1007/s10791-006-9019-z
25. Xu J, Li H (2007) Adarank: aboosting algorithm for information retrieval. In: Proceedings of the 30th annual international ACM SIGIR conference on research and development in information retrieval, SIGIR '07, ACM, New York, NY, USA. pp 391–398, ISBN 978-1-59593-597-7. https://doi.org/10.1145/1277741.1277809
26. Islam MA (2013) Rankgpes: learning to rank for information retrieval using a hybrid genetic programming with evolutionary strategies
27. TechTarget Contributor. What is evolutionary computation?: Definition from techtarget, Jul 2019. URL https://www.techtarget.com/whatis/definition/evolutionary-computation
28. Hofmann Katja, Whiteson Shimon, de Rijke Maarten (2013) Balancing exploration and exploitation in listwise and pairwise online learning to rank for information retrieval. Inf Retr 16(1):63–90
29. Schuth A, Hofmann K, Whiteson S, de Rijke M (2013) Lerot: an online learning to rank framework. In: Proceedings of the 2013 workshop on Living labs for information retrieval evaluation CIKM 2013, San Francisco, California, USA, November 1, pp 23–26, 2013. https://doi.org/10.1145/2513150.2513162
30. Qingyao Ai, Tao Yang, Huazheng Wang, Jiaxin Mao (2021) Unbiased learning to rank: online or offline? ACM Trans Inf Syst. https://doi.org/10.1145/3439861
31. Ibrahim OA (2017) Evolutionary algorithms and machine learning techniques for information retrieval. PhD thesis, University of Nottingham
32. Oosterhuis HR (2020) Learning from user interactions with rankings: a unification of the field. PhD thesis, University of Amsterdam
33. Vardasbi A, de Rijke M, Markov I (2021) Mixture-based correction for position and trust bias in counterfactual learning to rank. In: Proceedings of the 30th ACM international conference on information & knowledge management, CIKM '21, pp 1869-1878, New York, NY, USA. Association for Computing Machinery. ISBN 9781450384469. https://doi.org/10.1145/3459637.3482275
34. Zheng Y, Wu J, Wang B (2021) Clgbo: an algorithm for constructing highly robust coding sets for dna storage. Front. Genet, 12. ISSN 1664-8021. https://doi.org/10.3389/fgene.2021.644945., URL https://www.frontiersin.org/articles/10.3389/fgene.2021.644945
35. Qin T, Liu T-Y (2013) Introducing LETOR 4.0 datasets. CoRR. arxiv:abs/1306.2597. http://arxiv.org/abs/1306.2597
36. Chapelle O, Chang Y (2010) Yahoo! learning to rank challenge overview. In: Proceedings of the Yahoo! learning to rank challenge, held at ICML 2010, Haifa, Israel, June 25, pp 1–24, 2011. http://www.jmlr.org/proceedings/papers/v14/chapelle11a.html
37. Liu T-Y (2011) Chapter the LETOR datasets. In: Learning to rank for information retrieval, Springer, Berlin, Heidelberg. pp 133–143. ISBN 978-3-642-14267-3. https://doi.org/10.1007/978-3-642-14267-3_10