An efficient information retrieval system using evolutionary algorithms

network

MDPI

Article

An Efficient Information Retrieval System Using Evolutionary Algorithms

Doaa N. Mhawi [1], Haider W. Oleiwi [2], Nagham H. Saeed [3,*] and Heba L. Al-Taie [4]

[1] Department of Computer Systems Techniques, Middle Technical University, Baghdad 10010, Iraq
[2] Department of Electronic and Electrical Engineering, Brunel University, London UB8 3PH, UK
[3] School of Computing and Engineering, University of West London, London W5 5RF, UK
[4] Information and Communication Technologies Center, Ministry of Construction and Housing, Baghdad 10010, Iraq
* Correspondence: nagham.saeed@uwl.ac.uk

Abstract: When it comes to web search, information retrieval (IR) represents a critical technique as web pages have been increasingly growing. However, web users face major problems; unrelated user query retrieved documents (i.e., low precision), a lack of relevant document retrieval (i.e., low recall), acceptable retrieval time, and minimum storage space. This paper proposed a novel advanced document-indexing method (ADIM) with an integrated evolutionary algorithm. The proposed IRS includes three main stages; the first stage (i.e., the advanced documents indexing method) is preprocessing, which consists of two steps: dataset documents reading and advanced documents indexing method (ADIM), resulting in a set of two tables. The second stage is the query searching algorithm to produce a set of words or keywords and the related documents retrieving. The third stage (i.e., the searching algorithm) consists of two steps. The modified genetic algorithm (MGA) proposed new fitness functions using a cross-point operator with dynamic length chromosomes with the adaptive function of the culture algorithm (CA). The proposed system ranks the most relevant documents to the user query by adding a simple parameter ($\propto$) to the fitness function to guarantee the convergence solution, retrieving the most relevant user's document by integrating MGA with the CA algorithm to achieve the best accuracy. This system was simulated using a free dataset called WebKb containing Worldwide Webpages of computer science departments at multiple universities. The dataset is composed of 8280 HTML-programed semi-structured documents. Experimental results and evaluation measurements showed 100% average precision with 98.5236% average recall for 50 test queries, while the average response time was 00.46.74.78 milliseconds with 18.8 MB memory space for document indexing. The proposed work outperforms all the literature, comparatively, representing a remarkable leap in the studied field.

Keywords: culture algorithm; document indexing method; evolutionary algorithm; genetic algorithm; information retrieval systems

Citation: Mhawi, D.N.; Oleiwi, H.W.; Saeed, N. H.; Al-Taie, H.L. An Efficient Information Retrieval System Using Evolutionary Algorithms. *Network* **2022**, *2*, 583–605. https://doi.org/10.3390/network2040034

Academic Editors: Hakim Mellah and Filippo Malandra

Received: 31 August 2022
Accepted: 26 October 2022
Published: 28 October 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

1. Introduction

The field of information retrieval (IR) was born in the 1950s out of this necessity. Over the last forty years, the field has matured considerably. Several IR systems are used on an everyday basis by a wide variety of users. IR finds textual documents that satisfy a user's information needs from within a substantial number of documents, which are commonly stored on computers. Nowadays, millions of people engage in IR when they use search engines for the web and use their e-mails. Additionally, IRS covers other kinds of data and information problems not specified beyond that specified in the core definition. Data that do not have a clear and semantically explicit meaning are called unstructured data. This is the opposite of structured data, for example, the relational database of the sort firms that used to maintain product inventories and staff records. Practically, no data are

truly "unstructured." This is true of all text data if you count the concealed linguistic structure of human languages. Most text, which is represented in documents by explicit markup (such as the coding underlying web pages), has structure, such as headings, paragraphs, and footnotes. For "semi-structured" information such as a document where the title contains Java and the body contains threading, the IRS is also used to facilitate the search for information type [1]. One goal of the IRS, which has special importance, is to provide the requirements of users with the most relevant documents. Different techniques and models were designed to achieve this important goal [2]. One of the main tasks in IR is classification, which is a characteristic that is suitable for machine learning. Learning algorithms use examples, attributes, and values, which IRSs supply in great quantities. This system includes three areas: symbolic learning, neural networks, and evolutionary computation-based algorithms [3,4]. Some of the most widely used learning-based IR models are evolutionary computing algorithms such as genetic algorithms (GA) [3].

In addition, a universal repository of knowledge and culture in the digital world has allowed the direct sharing of ideas and information at an unpredictable rate. So, there is a need to access the digital world's data in the form of documents. These documents are useful for sharing information with every user, which is considered information retrieval. It is a computerized process of producing relevance-ranked list documents based on an inquirer's request [1]. Therefore, in recent years, using textual datasets to speed up the execution of critical tasks has been part of many people's and organizations' daily routines; most of the datasets include a massive collection of documents from a variety of sources, where datasets could be research papers, articles, news, digital libraries, books, messages, e-mails, or web pages. Due to the continuous development of information technology, the huge amount of information (datasets) is growing. Hence, finding the most satisfactory information from databases is increasingly challenging [2]. Data organization is required to conduct data update and query operations effectively. Indexing is one of the adopted methods, where different indexing techniques are being explored for the content [3]. Indexing techniques are essential to improve the dataset's performance and security. In contrast, information retrieval (IR) indices have a variety of drawbacks, including huge index sizes, the inability to accumulate an output search, and potential security risks.

There are more motivations for this paper to develop the proposed system, i.e., generally, two main problems in IRSs are still encountered by the web user when trying to retrieve documents, which are related to the user query; one of these is that many of the retrieved documents are highly ranked and are not related to the request of the user. Additionally, the main information retrieval problem is determining the document's relevance to the user's requirements. Practically, this issue is considered a ranking problem that needs to be solved based on the matching between all documents and the user query dealing with information retrieval [5,6]. The second problem is that many related documents are found in the dataset but are not retrieved [7–13]. The relevant documents can be found by generating an optimal query [14–19]. An initial query is used with random or estimated weights, and new queries are generated based on the modification of the new weights. Based on the identified techniques, the new retrieval model is evaluated by comparing it with some existing models in a controlled environment. Thus, these approaches are still insufficient [20–23]. Therefore, we propose the advanced document indexing method (ADIM) as a preprocessing stage that takes less time to build and produces a lower index size. After the modified genetic algorithm (MGA) was integrated with the culture algorithm (CA) for the first time, these two algorithms were adapted and developed to work in IRSs to retrieve relevant documents for user queries.

Thus, the new indexing method was created on our proposed methods that can deal with huge datasets and respond to queries in near-linear time with little I/O overhead.

To the best of the authors' knowledge, targeted results have not yet been achieved. However, the novel work of this paper uniquely contributes to solving these problems by:

- Proposing a novel indexing technique called the advanced document indexing method (ADIM) applied to large IRS-indexed files joined with modified GA and CA for retrieving relevant documents to the user queries;
- Reducing the amount of storage required for the produced ADIM;
- Modifying genetic algorithm (MGA) and integrating with culture algorithm (CA) to retrieve relevant documents.

The remainder of the paper structured as follows: Section 2 includes a background and literature review. Section 3 explains the details of the proposed system (methodology). Section 4 illustrates the proposed system implementation. Section 5 describes the discussion and analysis of the conducted results. Finally, Section 6 summarizes the conclusions and recommendations for further study.

2. Background and Literature Review

IRS is the process whereby the user information needs an actual list of citations converted into documents in storage containing information that is useful to the user. IRS stores and manages the information on documents. The system serves the users in finding the useful information they need; the IRS explicitly does not return the information; it returns the location and existence of the document that information might contain instead. IRS includes the relevant documents that satisfy the user information and does not include irrelevant documents [24–26]. Figure 1 depicts the general structure of the IRS.

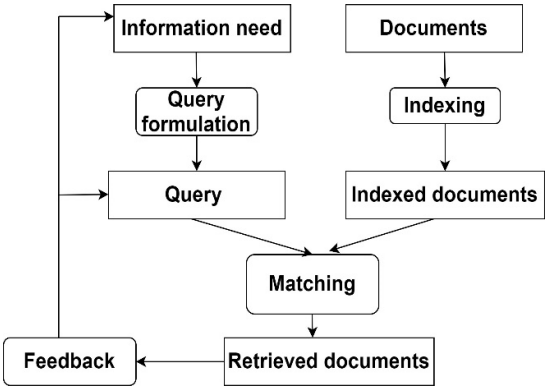

Figure 1. The general structure of the IRS.

In Figure 1, the user and system communicate with each other using respective queries, retrieving the set of documents. The most natural form of communication is used to communicate with each other for the information needed; such a natural communication method is called a request. In the automatic query, it takes the input as a request and gives the output as the initial query. Based on the initial query, some or all words in the request are converted to query terms by a trivial algorithm. Relevance feedback inputs the initial query to some retrieved relevant or irrelevant documents to output a successive query. The next subsections describe the IRS' techniques and methods.

2.1. Information Retrieval System (IRS) Concept

2.1.1. IRSs Models

The development of the documents and user query representing information as well as retrieval method or processes are all described by the information retrieval (IR) model. Three models make up the fundamental IR models (i.e., Boolean, vector space, and probabilistic) [27–29]. Table 1 depicts each of them with limitations.

Table 1. General description and limitations of IRS models.

Model Type	General Description	Limitations
Boolean	It uses a theory set, which is Boolean algebra. It has three elements (i.e., the NOT, the OR, and the AND) to form a query.	• It fails to rank the result list of retrieved documents. • Each document is linked to a certain group of words or keywords. • User queries can also be expressed as keyword phrases with AND, OR, or NOT separating them. • The Boolean function classifies a document based on its relevance.
Vector Space	It aims to order documents according to how closely each one resembles the user query. Documents and user queries are represented as a vector, whereas the angle between the two vectors is calculated using a cosine function. The vector space model has been introducing a term-weight scheme named the tf-idf weighting.	• It is required to compute weights. • Weights include a term frequency factor (tf) that counts the number of times a term appears in a document or user-query text as well as an inverse document frequency factor (idf) that counts the opposite number of documents that a term appears in.
Probabilistic	It is initiated to order documents according to how likely they are to be relevant given a user query. Vectors d and q, which are binary vectors, represent both documents and user queries.	• For the probabilistic model, index term weight variables are all binary.

2.1.2. Indexed Techniques

There are several popular indexing methods for IR such as signature files and inverted indexes. Table 2 depicts the description and limitations of these techniques.

Table 2. General description and limitations of indexed techniques.

Indexed Technique	General Description	Limitations
Signature File	Compared to the original file, it is significantly smaller. It also has a greater search rate.	• It is used only in security systems.
Inverted Index	Every document is composed of a list, which depicts the contents of the document for retrieval purposes. Fast retrieval is obtained if one can invert those keywords. All the reference words are stored alphabetically in a file called an index file. For each keyword, a list can be kept of pointers to the characterized documents in the postings file.	• It is used by all commercial systems and can be used in large spaces in memory but required a high execution time.

2.2. Evolutionary Models (Search Techniques) Overview

There are various search methods (i.e., linear, brute force, and binary) [30,31]. The search techniques and their disadvantages are described in Table 3.

Table 3. General description and disadvantages of searching techniques.

Searching Technique	General Description	Disadvantages
Linear	It is a fundamental method for discovering a certain word or keyword from a list of words or array that sequentially and individually verifies each element's presence. It is the least complicated technique.	• It is very slow, particularly when dealing with an ordered list.
Brute Force	It is a well-known technique, itemizing all potential participants to the resolution, determining whether each participant reveals problem statements.	• Despite its simplicity, it returns a solution when found.
Binary	It finds the position of a particular input value (i.e., the search key) within an array sorted by some key value. The given array needs to be arranged in an ascending or descending order. The middle element key value of the provided arranged array is compared to the search key value using this technique. When both	• The procedure repeats the procedure to the sub-array (left or right) of the main element when the search key value is smaller or greater than the key value of the element in between.

	keys' values matched, a matching item is discovered and indexed.	• If the left-over array is empty, the search key will not be in the array, and a specific string is provided, indicating that an array bit is missing.

2.2.1. Genetic Algorithms (GA) Overview and Related Work

The group of mathematical models known as GA is based on the ideas of natural selection and evolution. This heuristic is frequently employed to produce helpful optimization and look for answers to issues. GAs build chromosomes using selection factors, recombination, and mutations to develop chromosomes from selected problems using chromosome-like data.

Typically, a randomized set of chromosomes is used to start the GA process. These chromosomes symbolize a challenge that needs to be overcome. Depending on the characteristics of the problems, each chromosome's positions are re-coded with numerical, alphabetic, or bit values. The positions denote genes; they change randomly throughout the development process. A population is a group of chromosomes present during the evolutionary stage. The evaluation function determines each chromosome's validation. There are two main considerations while evaluating something. In order to imitate population transformation and natural breeding, the crossover is performed. Chromosomal selection favors the fittest chromosomes for survival and synthesis. In the form of partial functions or by utilizing various chromosome-coding schemes created expressly for a given issue, GAs can virtually mimic any type of limitation. Figure 2 depicts a simple GA structure.

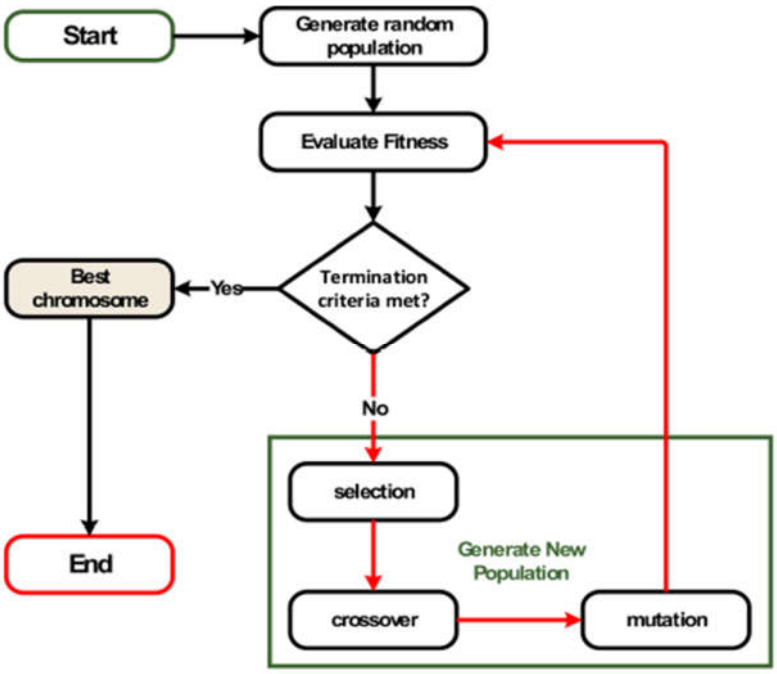

Figure 2. Genetic algorithm simple structure.

Many publications have introduced advanced interactive genetic algorithms (IGA) to improve IR.

Lee et al. [32] presented sparse fitness evaluation with an interactive genetic algorithm (IGA) to reduce the user's burden. Clustering was adopted to split the population into multiple subpopulations. A representative individual was selected from a subpopulation, where all fitness values of the remaining individuals were determined based on

the representative's value and distance. Lee et al. applied sparse fitness using 100 population sizes and 10 clusters.

A different IGA using a paired comparison (PC-IGA) was implemented by Watanabe et al. [33]. It permitted the required user to compare two individuals and select the best. The selected individual still competes until the best individual is obtained. The experimental results reached 7000 generations with different values of fitness function (FF).

Whereas IGA based on fuzzy logic was suggested by Sun et al. [34], the paper presents individuals' fuzzy and stochastic fitness for replacing the individual user's evaluation. It reduces users' burden by using a fuzzy number for fitness allocation. Furthermore, Sun et al. suggested building a surrogate model to evaluate individual fitness to improve IEC or to replace user evaluation with different ranking distances (i.e., 10 and 20), and different values of FF.

Additionally, Wang et al. [35] proposed the IGA using the support vector machine, which is used to construct the classifier taking advantage of the initial user's selected examples that method reduced user fatigue when using a 250 population size. The experimental results reached 94% recall, and the time of execution took many hours.

Moreover, a model of adaptive learning evaluation to assess beauty in the evolutionary art system was proposed by Li et al. [36]. It managed to extract specific features from the evolutionary images and real paintings. Furthermore, an accurate learning method was selected, training these features to establish a model. Another strategy was adopted for modifying evolutionary operators in IEC. The population size was 100, whereas the mutation rate was 0.5 with a high error rate.

2.2.2. Culture Algorithms (CA) Overview

Another evolutionary-based method is called a "cultural algorithm" (CA), which uses culture as a motivation to store related data that may be accessed by everyone in the population throughout several generations [37–39]. The culture may be seen as a dynamic source of information that shapes the social norms that diverse groups of people adhere to. Culture, like human civilizations, evolves with time, yet it offers a starting point for analyzing and recording a person's behavior within a community. CAs were created to simulate how the cultural component changes with time as it learns and gains knowledge. The belief space serves as a knowledge conduit between each generation that is evolving in CAs, which can be seen as an extension of genetic algorithms.

According to this methodology, CAs can be applied in numerous fields to lead self-adaptation activity across evolution systems. CAs are used as one of the specific models for evolution computations. Re-engineering of the commercial rule expert systems was carried out using CAs, and the knowledge discovery systems utilizing decision trees adopted a top-down strategy for networks' complexity reduction and performance enhancement. Due to the inherited complexity of real-word issues, semantic networks are being used to depict the numerous relations that occur within various problem contexts [37,38]. Figure 3 depicts the CA components.

To the best of the authors' knowledge, none of the previous research has investigated using the genetic algorithm and culture algorithm together in IRS.

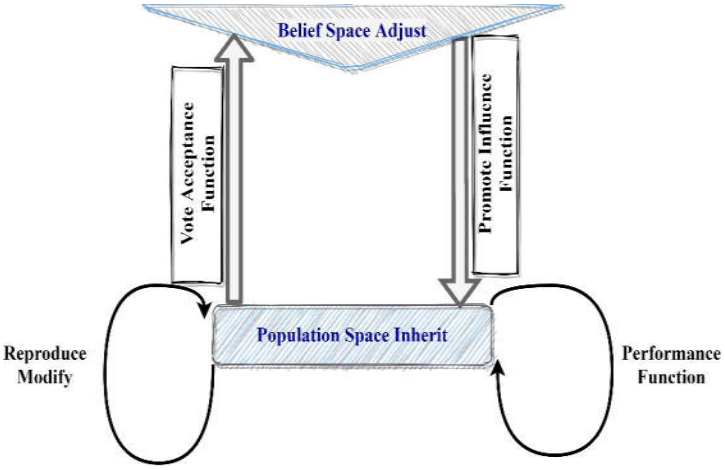

Figure 3. Components of culture algorithm.

3. Methodology

The proposed IRS includes three stages: stage 1 represents an advanced document indexing method (ADIM) used to prepare the WebKb dataset to maintain the high performance of IRS (the dataset requires preprocessing in order for the algorithm to work efficiently on accurate and reliable data), the second stage is query search processing, and the final stage is the evolutionary algorithms (i.e., genetic algorithm and culture algorithm) as an integration of two of the ML techniques. GAs are used to solve an optimization problem and are applied to reduce the overhead during the classification. CAs are also used for classification by applying the evaluation stage (i.e., precision, recall, and accuracy). The next subsections explain these stages.

3.1. Advanced Document Indexing Method (ADIM) Stage

This stage is composed of two main steps: WebKb dataset reading and ADIM. The main steps of ADMI are shown in the flowchart in Figure 4.

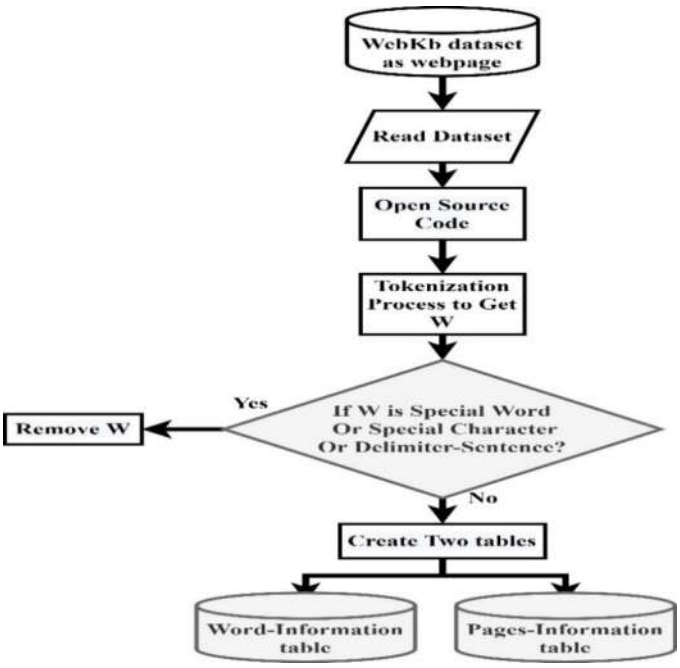

Figure 4. General flowchart of advanced document indexing method (ADIM).

3.1.1. WebKb Dataset Reading

This dataset contains worldwide webpages obtained from computer science departments at several universities. WebKb is composed of 8280 HTML-programed semi-structured documents. These documents contain seven directories: department, students, staff, faculty, projects, courses, and others, as shown in Table 4. The individual directory includes five classes with the university names as shown in Table 5.

Table 4. Directory names of WebKb dataset with a number of documents.

Directory Name	Number of Documents
Departments	1641
Students	181
Faculty	930
Courses	1124
Projects	3763
Staffs	504
Others	137
Total	8280

Table 5. Directory contents.

University Name	Number of Documents
Cornell	867
Texas	827
Washington	1263
Misc.	1205
Wisconsin	4120

3.1.2. ADIM

The search engines store indices of all the available documents in the documents index. It proposes a method for document indexing (i.e., ADIM), as described in Algorithm 1, that reduces the required memory storage space during query processing.

Document indexing fetches the documents' source codes, preprocessing them to create two tables: page information and word information. Page information is included in the page information table including ID, p name, word count, and total-tags weight. The word information table includes the related information of the words in terms of words and pages list.

Algorithm 1. Advanced Document Indexing Method

Begin

1. Read Dataset (i.e., WebKb) and open-source code of each document.
2. For each open document (i) do:
   Pages-information.id table = id(doc.(i))//Pages-information.id table represent directory, university, and page codes.
   Pages-information. Pagename table = Pagename. (doc.(i)).
3. While not EOF (open doc.(i)) do:
   Tokenization (Open doc.(i))//is a process of extracting the words (W).
   Retrieve (W).
   If W is Special-character or stop-word or Sentence-Delimiter, then
   Remove the (W).
   Else
   Pages-information. Total_weight table = total_weight(doc.(i),
   Pages-information. Total count-word table = summation (W),
   Words-information. Word table = W,
   Words-information. Pages-list table = all web page contains the same W,
   End If
4. End While

5. pages-information table (id, p-name, total-weight, and total-count-word),
6. words-information table (word, pages-list),
7. End For
8. End

Algorithm 1 starts by creating open-source code for each webpage in the Web Kb dataset and creates two tables; the first table is the Pages-information id table. This table is related to taking the directory, university, and page code from the dataset to facilitate access. Thereafter, it creates Pages-information. The page name table contains the name of each page. While not at the end of the document, the tokenization process is performed by extracting the words (W) from the document and removing them if this word is a special character or stop word or sentence delimiter, otherwise, the total weight and count of words for each web page table is computed. Lastly, a second table of word information is created. The word table implies words, and all web pages contain the same W. Hence, the output of this algorithm is these two tables.

The proposed documents indexing runs the following steps:

- The special-word table is constructed by removing (stop words, special characters, and sentence delimiters).
- The HTML tag information table is constructed by removing preferred tags (i.e., HTML, head, sub-headers (h1, h2, h3), body), and generating the weight of the removed tags. Table 6 depicts documents with tags; a document tag represents a specific importance level. It contains the related essential information of the requested user query.

Table 6. Document-tags dataset.

Tag-name	Weight	How Many Tags Occur on Page	Description
title	6	appear only one time and not repeated	It is the most important tag because contains terms near to the request of the UQ.
header & sub-header (h1, h2, and h3)	5	appear n times	words found within the tags deliver structural information.
An anchor	4	appear n times	This tag contains a word this word points to another word or link.
Italic (I) & bold (B)	3	appear n times	gives the descriptions of the document and the score.
body	1	appear only one time and not repeated	less important tag that contains the plain text.

In Table 6, the most important tags are given a high weight to retrieve the relevant documents in the shortest amount of time. Therefore, the title is a significant tag as it contains terms near to the request of the UQ, appears only one time, and is not repeated; hence, the weight is six, the weight of each header and sub-header (h1, h2, and h3) is five, the weight is four for an anchor tag, three for italics (I) and bold (B), and one for the body.

Then, the page-information table is created to find the total-tags-weight, given by Equation (1).

$$\text{Total-tag-weight} = \text{Weight (W) tag} + \sum \text{Weight (W) tag} \quad (1)$$

where the weight (W) tag is the weighting word.

3.2. Query Search Processing Stage

The proposed system converts a string of user queries into a set of words stored in an array. It fetches the word and page list from the word-information table, saving them in an array called query words (QW). The information of each word in the database is

acquired, merging each word of the page list with the other page-list words of the entered query into one array called the ID list. It is used as the initial generation of GA; however, the ID list contains a repeated ID that is useful in the initial generation as the page contains the most frequent ID. This increases the population's probability. Algorithm 2 describes the main steps of the query search algorithm (QS).

Algorithm 2. Query Search Algorithm (QSA)

1. Begin
2. Initialization: QW [word, pages-list] = [], X [word] = [], ID-list [pages-list] = [], Input user query (UQ = S) \ \ where S is string.
3. Loop//For i = 0 to UQ length
4. Split the user query (string) into words.
5. X [W] = split (S)
6. Stop until UQ is completed.
7. Loop//For each word in X [W] Do
8. QW [word, pages-list] = for each W get word.
9. QW [word, pages-list] = pages-list from words-information table.
10. ID-list [pages-list] = merge Pages-list for each word with the other pages-list word of the entered query.
11. Stop until the criteria are met.
12. Return ID-List array.
13. End

Algorithm 2 begins with initialization lists for both QW, X, and UQ. It splits UQ into words and stores these words in the list of X. For each word in the X list, obtain the word and page list from the word-information table in the results, create QW (word, page list), and merge the page list for each word with the other page-list words of the entered query to obtain the ID list (page list).

3.3. Evolutionary Algorithm Stage

3.3.1. Modify Genetic Algorithm (MGA)

GA was modified and integrated with CA to be adaptable with IR for optimization problems. It has several parameters that are implemented in various methods. GA operators (i.e., initial generation, fitness function, parent selection, crossover, and mutation) are summarized in Algorithm 3. The following subsections describe the details of these operators.

Algorithm 3. GA- Processes

1. Begin
2. Encode the problem.
3. Initial Generation.//First generation Randomly create.
4. In the first generation evaluate the FF for each chromosome.
5. Loop
6. Following operators to create the next generation:
7. Selection operator;
8. crossover operator;
9. mutation operator.
10. stop until the criteria are met.
11. From the last generation, return the best chromosome's content.
12. End

A. Initial Generation

Each chromosome is represented as a set of pages using the page ID, which is a list of three integers, i.e., the university, directory, and page code. Figure 5 depicts this.

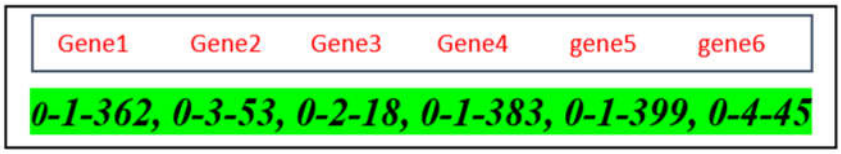

Figure 5. chromosome representation.

The chromosome length represents the number of shared pages between the query and QW, representing the ID list.

B. Fitness-Function

GA works to generate several generations before reaching the best solution or finding this solution. From the initial generation, these generations can be obtained by applying the operators of GA (i.e., selection, crossover, and mutation). Thus, the fitness function is used to assess the performance. FF is used to evaluate the relevance of documents to the user query. Documents are use the FF in two operators of GA; these two operators are selection and mutation. The proposed fitness function (FF) for this system is shown in Equation (2).

$$FF = \sum_{C=0}^{15-1} \sum_{g=0}^{sp-1} [(SPT(g) + 0.1) + WOT(g) + (PID(g) * 0.5) + \propto (g)] \quad (2)$$

where C is the chromosome, g is the gene (page), SP is the number of shared pages (pages that contain the words of the user query), SPT has shared pages (all pages contain words of user query together), WOT is 1–6 weights of tags (HTML, head, sub-header (h1, h2, h3), and body), PID is the page ID (each page contains the words of the user query together), and $\propto$ is a simple value that guarantees the remainder of all pages relevant to the user query at the first position of each chromosome, given by:

$$\propto = FF\ (page) - index\ (page) \quad (3)$$

All equation parameters are detailed as follows:

SPT represents several shared pages of words from user queries together. It can be checked within all pages from the initial generation to find which page contains the words of the user query together. Each appearance of query words together can increase 0.1 of the page fitness value determined by the experiment. In the following example, a chromosome consists of six shared pages; only three of them contain the words of the user query together. When the total weight and word count are computed from the page-information table, the FF value is 1100.

0-1-362, 0-3-53, 0-2-18, 0-1-383, 0-1-399, 0-4-45 1100

Check if the query found in these pages came together; if this condition is verified, it can be increased by 0.1 for each appearance of the query. Suppose the query words appear four times in these three pages so that the value of FF becomes computed as follows:

For the first appearance become: 1100 + 0.1 = 1100.1
For the second appearance becomes: 1100.1 + 0.1 = 1100.2
For the third appearance becomes: 1100.2 + 0.1 = 1100.3
For the third appearance becomes: 1100.3 + 0.1 = 1100.4

This is the final FF value of the chromosome, and the chromosome becomes:

0-1-362, 0-3-53, 0-2-18, 0-1-383, 0-1-399, 0-4-45 1100.4

WOT represents integers 1 to 6, the weight of query words appearing in tags. For each word of the user query, WOT assigns a weight to determine the tag of this query. For example:

If the query is the operating system and these query words come together, the query increases to six due to the query being in the tag title and the weight of this tag is six.

Moreover, if present in another page in the body, the weighting becomes one, as the weighting of the body is one, and accordingly for other queries.

Suppose the FF value of the page contains the query; the operating system is 1000, and this page contains this query in the title, and then this value is increased by six as follows:

FF = 1000 + 6 = 1006

Suppose this query appears on the body of the other page, then the FF value will be:

FF = 1000 + 1 = 1001

The WOT gives more importance to the query appearing in the title than to the query appearing in the body.

PID represents the page id and multiplies for each page, containing all queries together with a value of 0.5 determined by the experiment to ensure that the fitness of this page stays high.

C. Parent Selection

Parent selection is one of the three main operators of the GA, which is used to produce the next generation. Parent selection is controlled by FF. Better selection of parents of high quality ensures a higher probability to copy the best individual to the next generation to produce better offspring. There are several selection methods. One of these methods is to select parents randomly without any restrictions; this type is simple. However, this method has some disadvantages, such as that it allows irrelevant documents to be selected. The second method is called tournament selection; this method involves choosing a set of individuals randomly from the population. The third method is the truncation selection of this type with a threshold and fraction of the best individuals selected. The fourth method is the Genitor selection; this method works individual by individual during the selection of offspring for birth according to linear ranking and selection of the current worst individual to replace. The privilege of the simple random sampling method is over-proportional selection; it allows some weaker solutions to survive in the process of selection. The solutions may include some components, which could prove the usefulness of recombination. Furthermore, truncation selection may stick at local optima and cannot converge from initially selected chromosomes.

Finally, the elitism method is used in the MGA by the process of parent selection. This means the best individual copies in the next generation; this technique allows only strong individuals to participate in the process of the generation of solutions, preventing weak individuals. The best individuals have maximum fitness. This improves the GA performance increasingly.

D. Crossover Operator (Single-Point)

A simple and adequate single-point crossover is used in this proposed system to exchange the right part of the first chromosome with the right part of the second chromosome, starting with determining the cross-point (CP) value, which is calculated in Equation (4). CP splits the chromosome into two parts (left and right), then examines each gene on the first chromosome with the genes on the second chromosome. If this gene is the same one on the second, it stays on the first chromosome and does not exchange. However, if the gene is not found on the second chromosome, it can be exchanged with the second gene, and accordingly for the other genes. Ultimately, fitness values can be increased or decreased according to gene exchange. Algorithm 4 illustrates the proposed single-point crossover operation. Figure 6 depicts an example of the crossover operator.

$$\text{Cross-point (CP)} = \text{length of chromosome}/2 \quad (4)$$

Algorithm 4. Pseudo-Code of proposed single-point crossover operation

```
Begin
CP = Eq. (4).
Loop//For i = 0 to select. length-1
R1 = selected (R1) \ \ select random chromosome 1.
R2 = selected (R2) \ \ select random chromosome 2.
Swapping [Cost1 = population.chrom (R1). Fitness, Cost2 = population.chrom(R2). Fitness].
Swapping (R1, R2).
Newcost1 = get fitness (R1).
Newcost2 = get fitness (R2).
If (Newcost1 > cost1 or Newcost2 > cost2) then//Fitness function check.
Population. Chrom(R1). fitness = newcost1.
Population. ChromR2). fitness = newcost2.
Else
Swapping (R1, R2).
End if
End for
Return new R1, and R2.
End
```

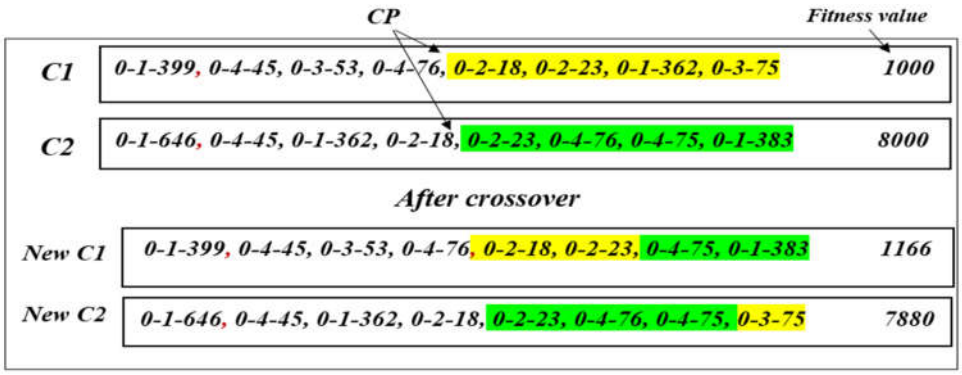

Figure 6. Crossover operator representation.

E. Mutation Operator

To avoid local convergence, the mutation is implemented with a tiny probability (1% in the presented work). In mutation, one gene is selected randomly from the generated offspring and replaced with one from the search space. Figure 7 demonstrates this operator.

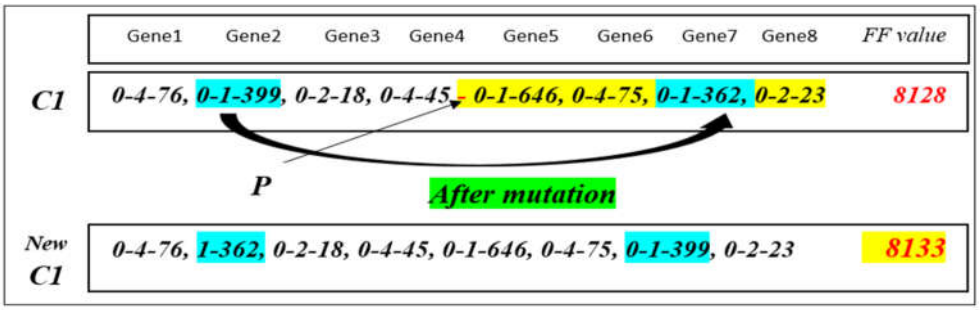

Figure 7. Mutation operator representation.

F. Stopping Criteria

The MGA operation is repetitive. Every iteration has a generation, consisting of a solution group of chromosomes. The iteration number was 15. Thus, the stopping criteria were either MGA with 15 iterations or lacking solution enhancement. The achieved enhancement was measured based on the predefined FF. Accordingly, it was lacking in accuracy enhancement. For enhancement, we assumed a threshold of 0.05 to measure the difference in accuracy between two consecutive generations; whenever the difference is less, the process stops.

3.3.2. Culture Algorithm (MCA)

After the MGA has generated the initial population, the CA uses the training set for the adaptation process to achieve high-performance results. Algorithm 5 depicts the details of this algorithm.

3.4. GA and CA Integration

We apply the ADIM (the preprocessing stage is explained later in Section 3.1), QSA (query search algorithm explained in Section 3.2), and evolutionary algorithms. MGA is used to select the best chromosome, as the data consist of high-dimensional features with seven different classes that are used to extract the prepared data. If the fitness function percentage is not acceptable, it affects the CA parameters, changing them and calculating a new accuracy. What is meant by not being accepted is that the accuracy is less than the threshold. For this work, the threshold was set to the minimum accuracy achieved by other researchers, which was 90.31% [40]. If the results are accepted, they are saved for future comparison with the results obtained from other iterations. The following step examines the iteration number. If it is greater than the threshold, then the result is stored, and the process stops; otherwise, the process returns to the first step. Figure 8 shows the flowchart of the presented integration system. In the proposed work, the GA at the first stage was for 15 generations. At each generation, the FF is examined. The proposed hybrid IDS algorithm is presented in Algorithm 5:

Algorithm 5. Integration between MGA, and CA

1. Begin
2. Read the WebKb dataset and apply ADIM Algorithm 1 to obtain two tables and split into two parts: training and testing set.
3. Loop
4. For each training and testing set
5. Create an initial generation.
6. End for
7. For i = 1 to iteration-number/*number of iteration is 15*/
8. Apply CA to the current training set with specific adaptations loop.
9. Evaluate the performance of the results using the FF.
10. Compute the accuracy, recall, precision, and f1-score
11. End for
12. If accuracy is accepted, then
13. Save the current result
14. Else
15. Go to step 6
16. End if
17. If the total number of iterations < threshold, then
18. Go to step 4
19. Else
20. Display the best result
21. End if

22. until the stop criteria are met.
23. Return the best final result
24. End

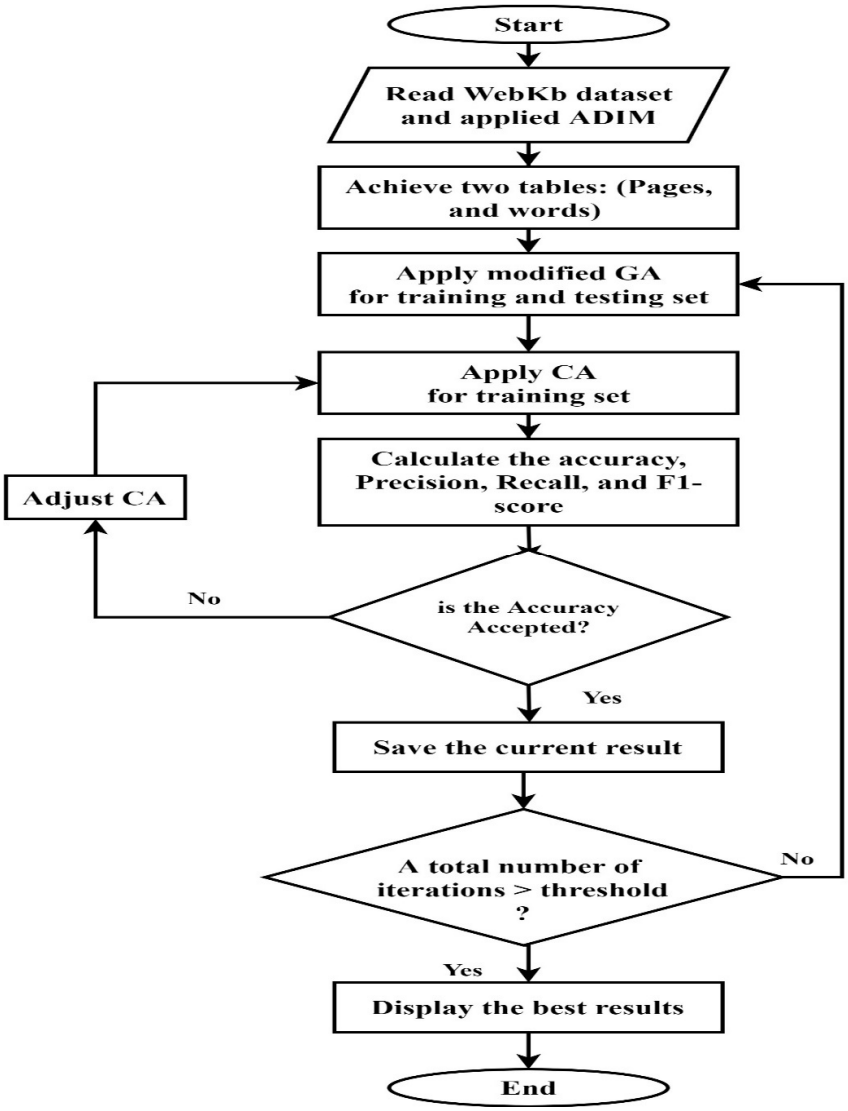

Figure 8. General flowchart of the presented integration system.

4. Implementation

The proposed system performance is evaluated using the following measures: accuracy, recall, precision, and F1-score. It is implemented by the software Visual Basic.Net 2019 and uses a Core i7 CPU and 64-bit OS Windows 11 as hardware. The following subsection explains the details of the implementation.

4.1. ADIM Experimental Results and Memory Efficiency

The proposed method indexes all meaningful words and adds the required information. This information is contained within two tables; the first table stores words and their page lists. The page list includes an ID list, which contains code on three entities (directory, university, and page). This ID reduces the searching time, retrieving the related documents faster. The memory space is reduced compared to the traditional method by

removing each word outside tags and each word attached with numbers or a special character, e.g., operating 565. The traditional method requires a large memory space of 2 bytes (2 × 8) for each entered read document. The total words in the dataset are 67,672; the required space is 8,965,186,560 MB; however, the proposed system requires only 18.8 MB to store data in memory, as shown in Figure 9.

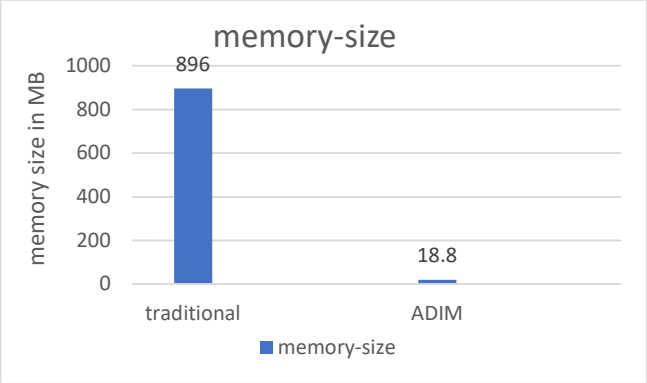

Figure 9. Memory space for document indexing.

Therefore, as the first contribution, ADIM reduces the time and memory space of the documents index by removing special words, delimiter sentences, and stop words.

4.2. Query Length Producer and Results

Since a one-word query is not meaningful, there are 50 queries with different lengths of two-to-four words prepared particularly to test the proposed system with a specified number of related documents. Table 7 depicts these queries with the length and number of related documents. The queries' numbers are used to evaluate the accuracy and performance of the proposed system to discover all the queries' related documents. Two main steps are applied: the first step (webpages intersection with the entered query words) finds and references all the documents that have all user query words; the second step (only shared page selection) filters all relevant documents to select only the documents that contain query words.

Table 7. List of a queries with length and number of related documents.

ID	Query	Query Length	No. of Related Doc.
1	WOODROW BLEDSOE	2	2
2	OPAL PROJECT	2	3
3	PASCAL PROGRAMMING	2	9
4	WERNER VOGELS	2	25
5	SOFTWARE TESTING TECHNIQUES	3	2
6	ANALYSIS CRYPTOGRAPHIC PROTOCOLS	3	1
7	LAUREN BRICKER	2	10
8	VLSI PLACEMENT ROUTING ALGORITHMS	4	1
9	DENNIS LEE	2	8
10	CAD VLSI RESEARCH GROUP	3	3
11	WAYNE OHLRICH	2	10
12	PAUL FRANKLIN	2	10
13	TED ROMER	2	13
14	MIKE DAHLIN	2	10
15	DAVID ZUCKERMAN	2	8
16	WEB OPERATING SYSTEMS	3	5
17	SOFTWARE ENGINEERING PROJECT	3	10
18	BENJAMIN KUIPERS	2	11

19	LORENZO ALVISI	2	12
20	COMPUTER COMMUNICATION NETWORKS	3	9
21	DANIEL WELD	2	7
22	CRAIG CHAMBERS	2	31
23	PROGRAMMING SOLUTIONS	2	3
24	CARL EBELING	2	36
25	STEVE HANKS	2	18
26	STEVEN TANIMOTO	2	9
27	PAUL YOUNG	2	3
28	EFFICIENT PARALLEL ALGORITHMS	3	8
29	NANCY LEVESON	2	14
30	DISCRETE STRUCTURES COMPUTER SCIENCE	4	5
31	ADVANCED PROGRAMMING LANGUAGES	3	15
32	OLVI MANGASARIAN	2	15
33	MIRON LIVNY	2	25
34	SCIENTIFIC COMPUTATION	2	30
35	PROGRAMMING LANGUAGES COMPILERS	3	19
36	ADVANCED DIGITAL DESIGN	3	7
37	CALTECH COMPUTER SCIENCE DEPARTMENT	4	5
38	OPAL PROJECT	2	3
39	MESH GENERATION-RELATED SOFTWARE	4	2
40	INTRODUCTORY COMPUTER PROGRAMMING	3	2
41	PROBLEM-SOLVING USING COMPUTERS	4	7
42	INTRODUCTION TO NATURAL LANGUAGE UNDERSTANDING	4	3
43	SYSTEM PROGRAMMER	2	3
44	JAVA-RELATED ITEMS	3	1
45	PROGRAMMING LANGUAGES IMPLEMENTA-TION	3	2
46	NEURAL NETWORKS INFORMATION	3	2
47	DIGITAL SYSTEMS DESIGN	3	25
48	NETWORKS DISTRIBUTED PROCESSING	3	1
49	NUMERICAL ANALYSIS COMPUTING	3	1
50	PARALLEL LANGUAGES COMPILERS	3	1

The last step of the proposed system is returning relevant documents to the user query with a high FF value in the last iteration (15 iterations) as an example using the following query:

Query: ANALYSIS CRYPTOGRAPHIC PROTOCOLS

This query contains four documents of shared pages, but only one relevant document contains all the words of the query together.

4.3. Modified GA Procedure and Results

A modified GA is used to integrate with a CA in the proposed IRS (the second contribution). The modified GA is based on a proposed FF, with a random selection mechanism, elitism mechanism to parent selection with proposed single-point crossover, and ordinary mutation with dynamic length to the chromosome. The optimal solution of the proposed system is achieved when setting the GA parameters to the values in Table 8.

Table 8. Modified genetic algorithm parameters.

ID	Parameters	Value
1	Population size	75 by experiment
2	Chromosome length (dynamic)	Dynamic (depending on the intersection between words of a query with web pages (shared pages)).
3	Number of iterations	15 y experiment

4.4. Experimental Result of Population Size and Discussion

The modified GA algorithm with different sizes of populations (25, 50, and 75) is evaluated by a recall measure using 20 queries. This list of queries is tested by 8280 documents of the dataset, with 15 numbers of iterations as shown in Table 9.

Table 9. Recall measure of the population by (25, 50, 75) size.

ID	Query	Recall Measures		
		25	50	75
1	WOODROW BLEDSOE	100%	100%	100%
2	OPAL PROJECT	33.33%	66.66%	100%
3	PASCAL PROGRAMMING	55.55%	66.66%	77.77%
4	WERNER VOGELS	92%	96%	96%
5	SOFTWARE TESTING TECHNIQUES	100%	100%	100%
6	ANALYSIS CRYPTOGRAPHIC PROTOCOLS	100%	100%	100%
7	LAUREN BRICKER	90%	100%	100%
8	VLSI PLACEMENT ROUTING ALGORITHMS	100%	100%	100%
9	DENNIS LEE	62.5%	62.5%	62.5%
10	CAD VLSI RESEARCH GROUP	66.66%	100%	100%
11	WAYNE OHLRICH	88.88%	88.88%	88.88%
12	PAUL FRANKLIN	50%	50%	70%
13	TED ROMER	76.69%	84.46%	84.46%
14	MIKE DAHLIN	70%	50%	70%
15	DAVID ZUCKERMAN	37.5%	50%	62.5%
16	WEB OPERATING SYSTEMS	80%	80%	80%
17	SOFTWARE ENGINEERING PROJECT	60%	90%	90%
18	BENJAMIN KUIPERS	72.72%	72.72%	81.81%
19	LORENZO ALVISI	100%	100%	100%
20	COMPUTER COMMUNICATION NETWORKS	55.55%	66.66%	77.77%

Table 10 shows the recall of these sizes while the precision measure is always 100%; we observed that the recall of 75 populations is better than the recall of the other two populations (25 and 50). The results are explained in Figure 10.

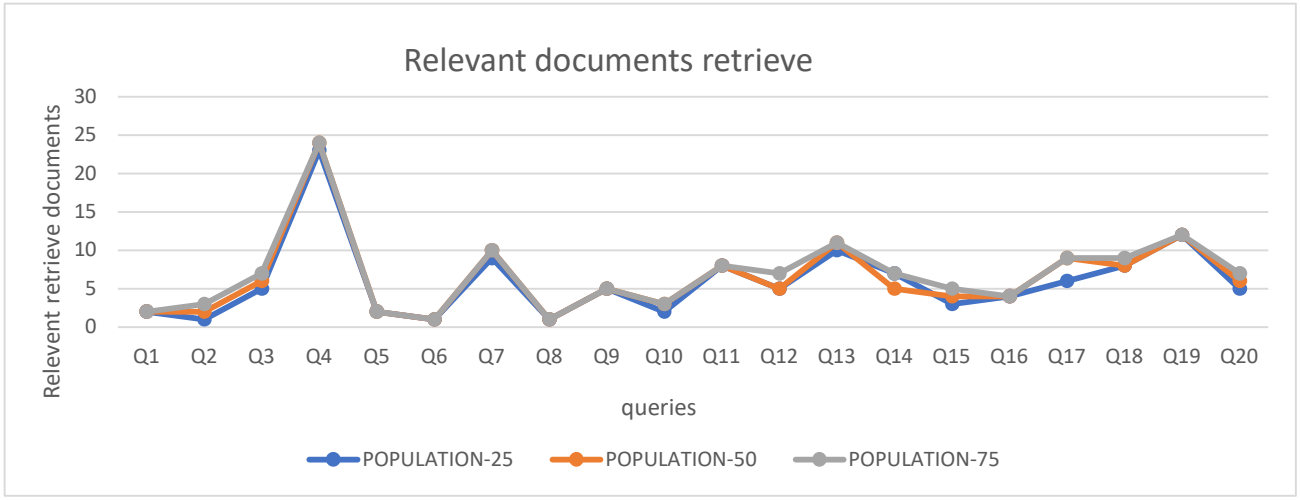

Figure 10. Recall to retrieval of relevant documents using (25, 50, 75) population size.

Figure 10 depicts the recall number of retrieved relevant documents in these sizes of populations. Figure 11 shows the accuracy of these populations.

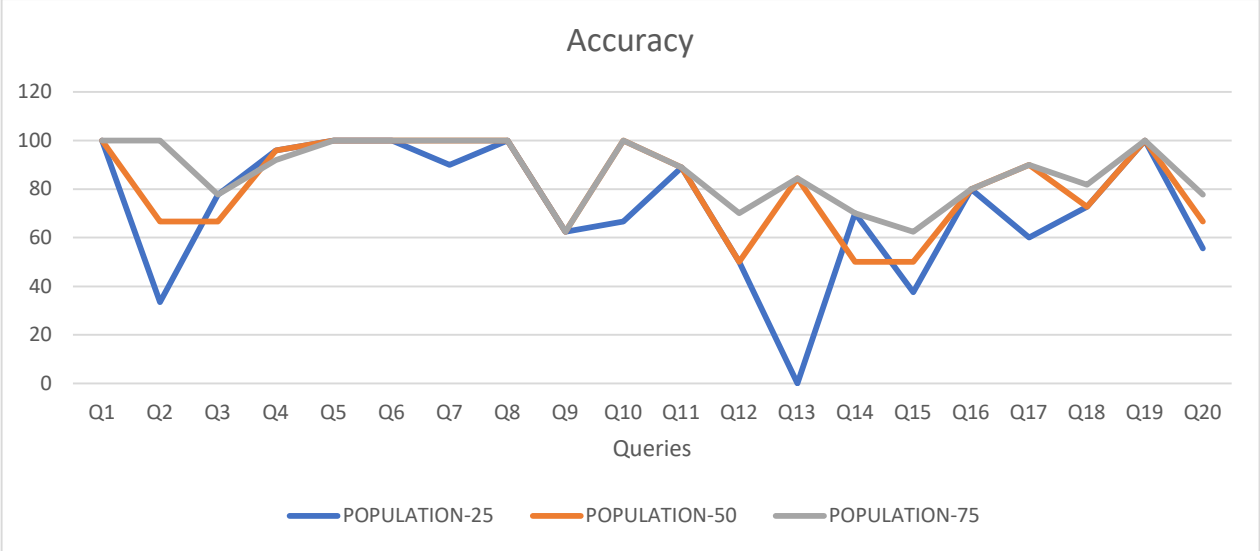

Figure 11. Accuracy of population size (25, 50, and 75).

The proposed system's effectiveness is evaluated based on the precision and recall measurements. Table 10 shows the proposed system test results of 50 different queries with different lengths.

Table 10. Experimental results of 50 queries.

ID	Query	Semantic Results	Recall	Precision
1	WOODROW BLEDSOE	2 of 2	100%	100%
2	OPAL PROJECT	3 of 3	100%	100%
3	PASCAL PROGRAMMING	9 of 9	100%	100%
4	WERNER VOGELS	25 of 25	100%	100%
5	SOFTWARE TESTING TECHNIQUES	2 of 2	100%	100%
6	ANALYSIS CRYPTOGRAPHIC PROTOCOLS	1 of 1	100%	100%
7	LAUREN BRICKER	10 of 10	100%	100%
8	VLSI PLACEMENT ROUTING ALGORITHMS	1 of 1	100%	100%
9	DENNIS LEE	8 of 8	100%	100%
10	CAD VLSI RESEARCH GROUP	3 of 3	100%	100%
11	WAYNE OHLRICH	9 of 9	100%	100%
12	PAUL FRANKLIN	9 of 10	90%	100%
13	TED ROMER	11 of 13	90%	100%
14	MIKE DAHLIN	10 of 10	100%	100%
15	DAVID ZUCKERMAN	8 of 8	100%	100%
16	WEB OPERATING SYSTEMS	5 of 5	100%	100%
17	SOFTWARE ENGINEERING PROJECT	9 of 10	90%	100%
18	BENJAMIN KUIPERS	10 of 11	90%	100%
19	LORENZO ALVISI	12 of 12	100%	100%
20	COMPUTER COMMUNICATION NETWORKS	9 of 9	100%	100%
21	DANIEL WELD	6 of 7	90%	100%
22	CRAIG CHAMBERS	30 of 31	90%	100%
23	PROGRAMMING SOLUTIONS	3 of 3	100%	100%
24	CARL EBELING	45 of 36	90%	100%
25	STEVE HANKS	17 of 18	90%	100%
26	STEVEN TANIMOTO	8 of 9	90%	100%
27	PAUL YOUNG	3 of 3	100%	100%
28	EFFICIENT PARALLEL ALGORITHMS	8 of 8	100%	100%
29	NANCY LEVESON	14 of 14	100%	100%

30	DISCRETE STRUCTURES COMPUTER SCIENCE	4 of 5	90%	100%
31	ADVANCED PROGRAMMING LANGUAGES	14 of 15	90%	100%
32	OLVI MANGASARIAN	15 of 15	100%	100%
33	MIRON LIVNY	24 of 25	96%	100%
34	SCIENTIFIC COMPUTATION	28 of 30	92%	100%
35	PROGRAMMING LANGUAGES COMPILERS	18 of 19	90%	100%
36	ADVANCED DIGITAL DESIGN	6 of 7	85.71%	100%
37	CALTECH COMPUTER SCIENCE DEPARTMENT	5 of 5	100%	100%
38	OPAL PROJECT	3 of 3	100%	100%
39	MESH GENERATION-RELATED SOFTWARE	2 of 2	100%	100%
40	INTRODUCTORY COMPUTER PROGRAMMING	2 of 2	100%	100%
41	PROBLEM-SOLVING USING COMPUTERS	7 of 7	100%	100%
42	INTRODUCTION TO NATURAL LANGUAGE UNDERSTANDING	3 of 3	100%	100%
43	SYSTEM PROGRAMMER	3 of 3	100%	100%
44	JAVA-RELATED ITEMS	1 of 1	100%	100%
45	PROGRAMMING LANGUAGES IMPLEMENTATION	2 of 2	100%	100%
46	NEURAL NETWORKS INFORMATION	2 of 2	100%	100%
47	DIGITAL SYSTEMS DESIGN	23 of 25	97%	100%
48	NETWORKS DISTRIBUTED PROCESSING	1 of 1	100%	100%
49	NUMERICAL ANALYSIS COMPUTING	1 of 1	100%	100%
50	PARALLEL LANGUAGES COMPILERS	1 of 1	100%	100%
	Average		98.5236%	100%

In Table 10, the results of the proposed system ADIM (MGA integrated with CA) show that the average value of recall is 98.5%, precision is 100%, and response time is 0.467 milliseconds. These results mean the proposed system is efficient in retrieving relevant documents to user queries by verifying two important principles of accuracy and fast retrieval time.

4.5. Comparison with Other Studies

A comparison study was carried out between the proposed IRS, traditional IR, and related work. Table 11 shows the comparison study. Table 11 shows the proposed IRS requires less memory storage and has a shorter response time for document indexing with high recall and precision values. These values prove the proposed system's effectiveness and performance.

Table 11. Comparison with traditional and other related work.

Item	Proposed	Ref. [20]	Traditional
Documents indexing method	ADIM	Enhance inverted index	Inverted index
Memory space for the doc. indexing	18.8 MB	100 MB	895.186 MB
Average Recall	98.5%	84%	N/A
Average of Precision	100%	86%	N/A
Response time in milliseconds	00.46.74.78 ms	N/A	Measured by minutes
Displaying the most relevant document	based on the fitness function	first 10 ranked documents	ranking method
Optimal solution convergence	15 iterations	22 iterations	N/A
Population size	75	125	N/A

5. Discussions and Analysis

Through several experiments, a set of evaluation measurements and comparisons with other related works were made of the following results:

(a) The proposed ADIM reduced the time of document indexing and reduced the memory space by using simple processes; these processes are removing special words and delimiter sentences and stopping word removal.

(b) The suitable number of queries is 50 queries to test the proposed system in terms of performance and accuracy. In addition, queries contain a different number of related documents (small, medium, and large).

(c) The suitable population size for the MGA is 75 to retrieve all the possible related documents to the user query or retrieved results close to the number of these relevant documents.

(d) CA applied the adaptation process when the result of accuracy is not sufficient, which ensures the best accuracy and high-performance system.

(e) A set of evaluation measurements and comparison with other related works made the results; the average value of recall is 98.5%, precision is 100%, and the response time is 0.46.7 milliseconds.

(f) GA operators work with a proposed FF, random selection mechanism, elitism mechanism to parent selection, single-point crossover, and ordinary mutation. The optimal solution of the proposed system is achieved with a population size of 75 at iteration 15.

(g) displaying the most relevant document to the user query requires no ranking step display because to find the optimal solution, the documents are ordered according to their fitness value; this process saves time in the ranking step and guarantees the most relevant document will be displayed first.

Figure 12 demonstrates the concise and concrete results when executing the system along with the visualization method.

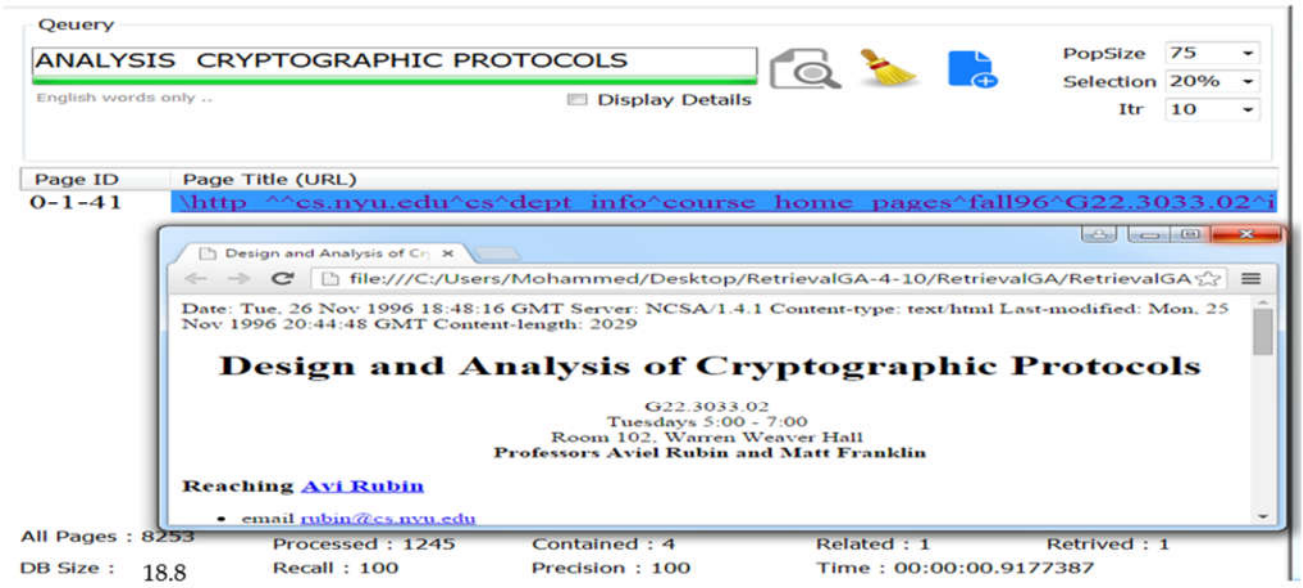

Figure 12. The general execution of ADIM-MGA with integrated CA.

6. Conclusions

MGA and CA were used in IRSs to solve two main problems that are still encountered by web users when trying to retrieve the relevant document. The first problem is the irrelevance of many highly ranked retrieved documents to the user's query, whereas the second problem is the irretrievable relevant documents in the dataset. The major aim of the paper is to develop a novel evolutionary-based IRS that retrieves user-relevant queries with the best performance of precision, recall, response time, and storage space as compared to all the previous systems, reducing the document indexing time and memory space. The experimental results reached average values of 100% precision, 98.5236% recall for all the test queries, 00.46.74.78 ms response time (i.e., the process of retrieval is fast), and 18.8 MB required memory space. Furthermore, the modified CA algorithm showed

the adaptability to work sufficiently with the developed operators, i.e., FF, random selection mechanism, elitism mechanism to parent selection with the proposed CP, and ordinary mutation technique. The optimal solution of the proposed system was achieved with a population size of 75 with 15 iterations, retrieving the most relevant documents to the user query without a ranking step. Moreover, this proposal saves the ranking step time and guarantees the most relevant documents to the user query as the documents are ordered according to their fitness value. We suggest designing another type of document indexing that can be employed with different types of documents without any preprocessing and to reduce the required memory space. We also suggest applying the proposed system to a larger dataset.

Author Contributions: Conceptualization, D.N.M.; Formal analysis, D.N.M., H.W.O., and H.L.A.-T.; Software, D.N.M.; Supervision, H.W.O., and N.H.S.; Writing – original draft, D.N.M.; Writing – review & editing, D.N.M., H.W.O., H.L.A.-T., and N.H.S.

Funding: This research has been discounted by Nagham H. Saeed.

Data Availability Statement: http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/.

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

References

1. Ceri, S.; Bozzon, A.; Brambilla, M.; Della Valle, E.; Fraternali, P.; Quarteroni, S. An Introduction to Information Retrieval. In *Web Information Retrieval*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 3–11.
2. Sarrouti, M.; Ouatik El Alaoui, S. A passage retrieval method based on probabilistic information retrieval and UMLS concepts in biomedical question answering. *J. Biomed. Inform.* **2017**, *68*, 96–103. https://doi.org/10.1016/j.jbi.2017.03.001.
3. Kantemirova, B.I.; Orlova, E.A.; Polunina, O.S.; Chernysheva, E.N.; Abdullaev, M.A.; Sychev, D.A. Pharmacogenetic bases of individual sensitivity and personalized administration of antiplatelet therapy in different ethnic groups. *Farmatsiya Farmakol.* **2020**, *8*, 392–404. https://doi.org/10.19163/2307-9266-2020-8-6-392-404.
4. Oleiwi, H.W.; Mhawi, D.N.; Al-Raweshidy, H. MLTs-ADCNs: Machine Learning Techniques for Anomaly Detection in Communication Networks. *IEEE Access* **2022**, *10*, 91006–91017. https://doi.org/10.1109/access.2022.3201869.
5. Zhang, S.; Yao, L.; Sun, A.; Tay, Y. Deep Learning Based Recommender System. *ACM Comput. Surv.* **2019**, *52*, 1–38. https://doi.org/10.1145/3285029.
6. Erritali, M.; Beni-Hssane, A.; Birjali, M.; Madani, Y. An Approach of Semantic Similarity Measure between Documents Based on Big Data. *Int. J. Electr. Comput. Eng.* **2016**, *6*, 2454–2461. https://doi.org/10.11591/ijece.v6i5.10853.
7. Kulzer, B. Wie profitieren Menschen mit Diabetes von Big Data und künstlicher Intelligenz? *Der Diabetol.* **2021**, *17*, 799–806. https://doi.org/10.1007/s11428-021-00818-9.
8. Han, Y.; Lang, Y.; Cheng, M.; Geng, Z.; Chen, G.; Xia, T. DTaxa: An actor–critic for automatic taxonomy induction. *Eng. Appl. Artif. Intell.* **2021**, *106*, 104501. https://doi.org/10.1016/j.engappai.2021.104501.
9. Vatansever, D.; Smallwood, J.; Jefferies, E. Varying demands for cognitive control reveals shared neural processes supporting semantic and episodic memory retrieval. *Nat. Commun.* **2021**, *12*, 1–11. https://doi.org/10.1038/s41467-021-22443-2.
10. Jabonete, D.S.; De Leon, M.M. Development of an Automatic Document to Digital Record Association Feature for a Cloud-Based Accounting Information System. In *Lecture Notes in Networks and Systems*; Springer: Berlin/Heidelberg, Germany, 2022; Volume 283.
11. Oleiwi, H.W.; Al-Taie, H.L.; Saeed, N.; Mhawi, D.N. A Comparative Investigation on Different QoS Mechanisms in Multi-Homed Networks. *Iraqi J. Ind. Res.* **2022**, *9*, 1–11. https://doi.org/10.53523/ijoirvol9i1id141.
12. Al-ufoq, N.; Company, N.A. In Proceedings of the The 3rd International Scientific Conference of Computer Sciences (3SCCS2021), Muscat, Oman, 14 August 2021; ISBN 9789922945552.
13. Oleiwi, H.W.; Saeed, N.; Al-Taie, H.L.; Nteesha, D. An Enhanced Interface Selectivity Technique to Improve the QoS for the Multi-homed Node. *Eng. Technol. J.* **2022**, *40*, 101–109. https://doi.org/10.30684/etj.2022.133066.1165.
14. Katoch, S.; Chauhan, S.S.; Kumar, V. A review on genetic algorithm: Past, present, and future. *Multimed. Tools Appl.* **2021**, *80*, 8091–8126. https://doi.org/10.1007/s11042-020-10139-6.
15. Kadhm, M.S. An Accurate Diabetes Prediction System Based on K-means Clustering and Proposed Classification Approach. *Int. J. Appl. Eng. Res.* **2018**, *13*, 4038–4041.
16. Alam, T.; Qamar, S.; Dixit, A.; Benaida, M. Genetic algorithm: Reviews, implementations and applications. *Int. J. Eng. Pedagog.* **2021**, *10*, 57–77. https://doi.org/10.3991/IJEP.V10I6.14567.
17. Velliangiri, S.; Karthikeyan, P.; Arul Xavier, V.M.; Baswaraj, D. Hybrid electro search with genetic algorithm for task scheduling in cloud computing. *Ain Shams Eng. J.* **2021**, *12*, 631–639. https://doi.org/10.1016/j.asej.2020.07.003.
18. Drachal, K.; Pawłowski, M. A review of the applications of genetic algorithms to forecasting prices of commodities. *Economies* **2021**, *9*, 6. https://doi.org/10.3390/economies9010006.

19. Delyová, I.; Frankovský, P.; Bocko, J.; Trebuňa, P.; Živčák, J.; Schürger, B.; Janigová, S. Sizing and topology optimization of trusses using genetic algorithm. *Materials* **2021**, *14*, 1–14. https://doi.org/10.3390/ma14040715.
20. Ren, J.; Wang, H.; Liu, T. Information retrieval based on knowledge-enhanced word embedding through dialog: A case study. *Int. J. Comput. Intell. Syst.* **2020**, *13*, 275–290. https://doi.org/10.2991/ijcis.d.200310.002.
21. Mhawi, D.N.; Hashem, S.H. Proposed Hybrid Correlation Feature Selection Forest Panalized Attribute Approach to advance IDSs. *Karbala Int. J. Mod. Sci.* **2021**, *7*, 405–420. https://doi.org/10.33640/2405-609X.3166.
22. Mhawi, D.N.; Aldallal, A. Advanced Feature-Selection-Based Hybrid Ensemble Learning Algorithms for Network Intrusion Detection Systems. *Symmetry* **2022**, *14*, 1461.
23. Oleiwi, H.W.; Saeed, N.; Al-taie, H.L.; Mhawi, D.N. Evaluation of Differentiated Services Policies in Multihomed Networks Based on an Interface-Selection Mechanism. *Sustainability* **2022**, *14*, 1–12. https://doi.org/10.3390/su142013235.
24. El-Bathy, N.; Azar, G.; El-Bathy, M.; Stein, G. Intelligent information retrieval lifecycle architecture based clustering genetic algorithm using SOA for modern medical industries. In Proceedings of the IEEE International Conference on Electro Information Technology, Mankato, MN, USA, 15–17 May 2011.
25. Zhang, P.; Gao, H.; Hu, Z.; Yang, M.; Song, D.; Wang, J.; Hou, Y.; Hu, B. A bias–variance evaluation framework for information retrieval systems. *Inf. Process. Manag.* **2022**, *59*, 102747. https://doi.org/10.1016/j.ipm.2021.102747.
26. Bhardwaj, S.; Sharma, S. An automated framework for incorporating fine-grained news data into S&P BSE SENSEX stock trading strategies. *Indian J. Sci. Technol.* **2016**, *9*, 97025. https://doi.org/10.17485/ijst/2016/v9i37/97025.
27. Wang, F.; Liu, J.; Wang, H. Sequential Text-Term Selection in Vector Space Models. *J. Bus. Econ. Stat.* **2021**, *39*, 82–97. https://doi.org/10.1080/07350015.2019.1634079.
28. Hassan, A.K.; Enteesha mhawi, D. Enhance Inverted Index Using in Information Retrieval. *Eng. Tech. J.* **2016**, *34*, 302–310.
29. Karim Abdul Hassan, A.; Enteesha mhawi, D. A Proposed Method for Documents Indexing. *Diyala J. Pure Sci.* **2017**, *13*, 43–56. https://doi.org/10.24237/djps.1302.144a.
30. El Guemmat, K.; Ouahabi, S. A literature review of indexing and searching techniques implementation in educational search engines. *Int. J. Inf. Commun. Technol. Educ.* **2018**, *14*, 72–83. https://doi.org/10.4018/IJICTE.2018040106.
31. Bukar, A.L.; Tan, C.W.; Said, D.M.; Dobi, A.M.; Ayop, R.; Alsharif, A. Energy management strategy and capacity planning of an autonomous microgrid: Performance comparison of metaheuristic optimization searching techniques. *Renew. Energy Focus* **2022**, *40*, 48–66. https://doi.org/10.1016/j.ref.2021.11.004.
32. Lee, J.Y.; Cho, S.B. Sparse fitness evaluation for reducing user burden in interactive genetic algorithm. In Proceedings of the IEEE International Conference on Fuzzy Systems, Seoul, Korea, 22–25 August 1999; Volume 2.
33. Shirakawa, M.; Arakawa, M. Multi-objective optimization system for plant layout design (3rd report, Interactive multi-objective optimization technique for pipe routing design). *J. Adv. Mech. Des. Syst. Manuf.* **2018**, *12*, JAMDSM0053. https://doi.org/10.1299/jamdsm.2018jamdsm0053.
34. Sun, X.; Gong, D. Surrogate model-assisted interactive genetic algorithms with individual's fuzzy and stochastic fitness. *J. Control Theory Appl.* **2010**, *8*, 189–199. https://doi.org/10.1007/s11768-010-8223-y.
35. Pal, S.K.; Bandyopadhyay, S.; Biswas, S. *Pattern Recognition and Machine Intelligence—First International Conference, PReMI 2005, Proceedings*; Springer: Berlin/Heidelberg, Germany, 2005; Volume 3776 LNCS, ISBN 3540305068.
36. Liaw, R.T. A cooperative coevolution framework for evolutionary learning and instance selection. *Swarm Evol. Comput.* **2021**, *62*, 100840. https://doi.org/10.1016/j.swevo.2021.100840.
37. Rychtyckyj, N.; Reynolds, R.G. Using Cultural Algorithms to Improve Knowledge Base. *IEEE Congr. Evol. Comput.* **1998**, *3*, 1405–1412.
38. Ohsaki, M. An input method using discrete fitness values for interactive GA. *J. Intell. Fuzzy Syst.* **1998**, *6*, 131–145.
39. Oleiwi, H.W.; Al-Raweshidy, H. SWIPT-Pairing Mechanism for Channel-Aware Cooperative H-NOMA in 6G Terahertz Communications. *Sensors* **2022**, *22*, 6200. https://doi.org/10.3390/s22166200.
40. Aldallal, A.; Alisa, F. Effective intrusion detection system to secure data in cloud using machine learning. *Symmetry* **2021**, *13*, 2306. https://doi.org/10.3390/sym13122306.