CHAPTER 5
THE IMPROVED INFORMATION ACCESS AND RETRIEVAL THROUGH GENETIC ALGORITHM (IIARGA)

5.1. Introduction

The research has proved that Genetic algorithm is a robust and powerful search mechanism for solving problems related to scientific and engineering applications, Information Retrieval being one of them (Cordón et al., 2003; Cecchini et al., 2008, Pathak et al., 2000). IIARGA or "Improved Information Access and Retrieval through Genetic Algorithm" is a GA-based algorithm. As the goal of IIARGA algorithm is to make the searching process effective and efficient for the user by retrieving more of relevant documents, the algorithm matches the indexed documents with the user query and retrieves the relevant set of documents and ranks them in the order of their relevance. In this chapter, we discuss the framework of the IIARGA which illustrates the functioning of the algorithm pictorially, followed by step-by-step algorithm. In the following sections the components of IIARGA are outlined, in conjunction with the description of GA operators being used in IIARGA to enhance its effectiveness.

5.2. The IIARGA Framework

Figure 5.1 describes the environment of IIARGA diagrammatically. There are three phases of IIARGA, namely preprocessing phase, evolutionary phase and ranking phase. The preprocessing phase involves performing a series of operations to convert the documents into a meaningful set of keywords and represent them in such a manner which aids the process of finding relevant documents. The evolutionary or GA phase is at the center of the algorithm, where improvement is performed. It is the prominent phase of the algorithm which helps in finding good answers to the query expressed by the user, by employing genetic algorithm. It makes use of the improved GA and then compares the indexed documents with the user query to retrieve the relevant set of documents. When the termination criteria is met, the optimal chromosome obtained represents the set of most relevant documents for that query. Then the documents are retrieved according to their ranks. The algorithm (Algorithm 5.1) details the working of IIARGA.
Algorithm 5.1: IIARGA

/* A GA-based algorithm to automatically generate optimised set of relevant documents*/

**Input:** Document Database
Set of Queries (Q)
Population size
Maximum no. of generations (Max_Gen)
Probability of crossover (CP)
Probability of mutation (MP)

**Output:** Set of documents Matching the query

**Begin**

//Preprocessing Phase
1. If (index not found) then
   1.1. Create index
2. End if
3. Load index in memory.
4. For each query q in Q, do the following:

//Evolutionary Phase
5. **Initialisation:**
   Create Initial population of chromosomes containing document descriptors as genes.
   Set no_of_generations =0.
   Set Initial Population as Current_Population
6. Repeat steps 6.1 to 6.6 while (no_of_generations<=Max_Gen) OR (fitness(best_phenotype) for successive 5 generations is same):
   6.1. Select two individuals.
   6.2. Apply CROSSOVER operator with a probability CP=0.8 to generate Temporary_population.
   6.3. Apply MUTATION operator on Temporary_population with a probability MP=0.7 to generate New_population.
   6.4. Calculate fitness for each chromosome using Customised fitness function DSF.
   6.5. Apply elitism to insert the fittest chromosome of Current_population into New_population.
   6.6. Select best_phenotype from the New_population.
7. End while

//RankingUnit
8. Apply reordering operator on the obtained optimal chromosome (best_phenotype).
9. Display the resultant documents to the user.

**End**
Figure 5.1: IIARGA Environment
5.2.1. Preprocessing Phase

**Document Representation**

The documents in their natural form cannot be evaluated directly through Information Retrieval. In order to perform retrieval, the documents and the query have to be represented in a computable form. The documents can either be of plain text, semi-structured or structured types. The IIARGA algorithm works on textual documents only. They are plaintext documents which are stored in XML format for preprocessing phase. The XML format for the documents is as depicted by Figure 5.2. Following section discusses an automatic indexing process that entails a set of text operations performed for IIARGA.

```xml
<DOC>
  <DOCNO>2</DOCNO>
  <YEAR>2011</YEAR>
  <TITLE>Restructuring of Global Economy — The HR Perspective</TITLE>
  <AUTHOR>Mr. Viresh Mathur, AVP, Corporate HR, Bal Krishna Tyres, Mumbai</AUTHOR>
  <TEXT>Global economic restructuring more particularly by the US, emerging economies of Asia and Latin America have resulted ...................... own insights and contribute to the key decisions apart from usual contribution in helping build business strategies. </TEXT>
  <KEYWORD>people management, chief HR officers, succession plan, competitive edge, organizational effectiveness</KEYWORD>
</DOC>
```

**Figure 5.2: Document representation for IIARGA**

**Indexing Process**

In order to improve the access to the documents, it is necessary that they are stored in an effective manner in terms of storage space and processing time. The indexing process is used to represent each document of the collection through a set of keywords or index terms (Ceri et al., 2013). Indexing, thus simply means representing text (the query and the document) as a set of terms whose combined semantic meaning is equivalent in some sense to the content of the original text (Sidiqui and Tiwary, 2005). The traditional Information Retrieval systems need to access the entire documents to decide whether it is relevant to a query. Also, they occupy larger storage space to store the indexes and take more time. It is more efficient to search the keywords of the query in the list and then retrieve the documents that index these terms.
These limitations can be fixed by using inverted indexing scheme (Singhal, 2001). The inverted index model was chosen for effective information access that allows retrieval of documents based on query terms. While creating inverted indexes, it is necessary to select the terms that need to be indexed for better representation of entire information within a document.

IIARGA uses Inverted index model for document representation, which is more efficient in terms of speed and processing, as it is fast to build and access. It can also store additional information about the keywords along with the document references like term frequency, etc. There is no need to search within the document, as documents can be identified by traversing the inverted list itself. The time complexity of the index construction is $O(T)$, where $T$ is the number of all terms (including duplicates) in the document collection (after pre-processing) (Liu, 2006).

The documents being parsed have been tokenized by extracting individual terms from the document, converting into lowercase and removing special terms like punctuation marks, digits and hyphens. Then each term has been evaluated. If it is a stop word, then it is ignored otherwise the term is stemmed to its root form to get the final index term. We have used the Inquery’s standard list of stop words (Huston and Croft, 2010; Allan et al., 2000). The stopwords are listed in Appendix A. The words have been stemmed using a Porter stemmer (Porter, 1997). The stemmed words are checked in the postings. If it is found, the document ID and its term frequency are added to the inverted list of the term. If the term is not found, a new node is created to represent the term. The flowchart (Figure 5.3) depicts the index creation process for IIARGA.

Two kinds of information are maintained with the indexes, local information and global information (Cummins and O’ Riordon, 2006a). The local information is the information about the terms within a document like raw term frequency, position, frequency of most occurred term, document length, and number of unique terms in a document. The global information is the information obtained through the entire collection like total number of documents in the database, vocabulary size,
number of unique terms and total terms in the collection, number of documents in which an index term was preset, and cumulative frequency of the term in the collection. Local information is collected at the time of index creation, while global information is obtained after the index creation is over. Both types of information are stored in the index, as they contribute to retrieval process.

As an example, say there are three documents as \(d_1, d_2, \) and \(d_3\):

\[
\begin{align*}
\text{d1} : & \text{ Genetic algorithm is robust.} \\
& 1 \ 2 \ 3 \ 4 \\
\text{d2} : & \text{ Search algorithm applications.} \\
& 1 \ 2 \ 3 \\
\text{d3} : & \text{ Genetic algorithm analyses the Genetic chromosome representation.} \\
& 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7
\end{align*}
\]

The numbers below each document are the offset position of each word. The vocabulary is the set: \{Genetic, algorithm, robust, applications, search, representation, analyses, chromosome\}

Stopwords “is” and “the” are removed, after stemming is applied, the terms set is: \{genet, algorithm, robust, applic, search, represent, analys, chromosom\}

The inverted index for these three documents are represented in Figure 5.4.

5.2.2. **Evolutionary Phase**

After inverted indexes are constructed for the given document collection, GA is applied to find relevant documents pertaining to the user query. The reason for choosing GA as an evolutionary technique for our algorithm has been discussed in the previous chapters. Now, we discuss the GA operators applied in IIARGA.
Figure 5.3: Flowchart for creating inverted index for IIARGA
5.2.2.1. Initial Population Creation

The iterative phase of evolutionary search process starts with an initial population of individuals. Each member of this population encodes a possible solution to a problem. The search space of the algorithm consists of the set of documents of the data collection from which the information is searched for. Each document is identified through a unique integer called document id which is used as the document reference number. The initial population of IIARGA consists of chromosomes, wherein each gene of a chromosome is an integer value depicting the document id (Al-Dallal and Abdulwahab, 2009). Thus, each chromosome contains a set of documents represented by document-ids as genes.

Initial population of IIARGA forms the first generation of the algorithm. It has been recognized that if the initial population to the algorithm is good, then the algorithm has a better possibility of finding a good solution (Zitzler et al., 2000; Radwan et al., 2006) and if the initial supply of building blocks is not large enough or good enough, then it would be difficult for the algorithm to find a good solution (Lobo and Lima, 2005). Building the initial population through random sampling of search space candidates is the simpler and a faster mechanism to create Initial Population (Radwan et al., 2006; Horng and Yeh, 2000; Pérez-Agüera, 2007), but the quality of chromosomes is low and takes more time to reach the optimal solution.
To improve upon the quality of solution, the initial population is seeded with potentially good or partial solutions of the problem (Snášel et al., 2009; Diaz-Gomez and Hougen, 2007). The random behavior of GA can be directed towards optimal solution by selecting comparatively healthier chromosomes for the initial population. Instead of pure random sampling, the initial population is created by adding filtration criteria to the randomly selected genes.

The randomly selected genes from the search space must meet two conditions in order to be added to the chromosome of IIARGA.

The first prerequisite for adding a gene or a document to the chromosome is of uniqueness of genes within a chromosome. A document is added to the chromosome only if it does not already exist in the chromosome (Al-Dallal and Shaker, 2009).

Given a chromosome:

\[ C = \{g_1, g_2, \ldots, g_L\} \]  \hspace{1cm} (5.1)

where \( g_i \) is a gene within the chromosome \( C \). A newly selected gene \( g_k \) is be added to the chromosome \( C \) only if \( g_k \notin C \).

The second condition applied on the chromosome is application of threshold value. A retrieval value threshold is set, if a document-query similarity value is below this threshold, the document is deemed irrelevant, and is not retrieved (Oren, 2002; Hmeidi et al., 1997; Yang and Korfhage, 1993). Setting a threshold value for the fitness value helps to measure the relativity of the document to the query more accurately by finding the degree of relevance of the document (Al-Dallal, 2012).

IIARGA uses this concept of setting the threshold value to discard irrelevant documents from each chromosome. If the fitness value of a document is greater than threshold then that document is added to the chromosome, else the document is discarded. The threshold value is set to 0.6.

Now 'Select' function becomes:

Given a document collection:

\[ V = \{d_1, d_2, \ldots, d_n\}, \]  \hspace{1cm} (5.2)

where \( d_i \): document descriptor number, \( i=\{1, 2, \ldots, n\} \)
Each document is composed of a set of indexed terms and is denoted by
\[ D = \{t_1, t_2, t_3, \ldots, t_m\} \]  
where \( t_j \): index terms, \( j = \{1, 2, \ldots, m\} \)

The fitness of a gene or document \( D \) is denoted by \( f(D) \). Then
\[
\text{Select}(D) = \{D \in V|D \cap C \text{ AND } f(D) > \text{ Threshold} \} \tag{5.4}
\]

The first condition is necessary to avoid duplicacy of same gene or
document-id in a chromosome. If this condition is overlooked, there may be a case
when the chromosome contains only one relevant document, but the fitness of the
chromosome is the highest. The second criterion of using threshold value is
important, since the value of the threshold affects the behaviour of the retrieval
operation. It helps the algorithm to discard irrelevant documents and give more
weightage to the relevant ones. The psuedocode for the creation of chromosomes of
the population is as given as Algorithm 5.2:

---

**Algorithm 5.2: Chromosome Creation Process of IIARGA**

**Given:** The length of chromosome is fixed.

**Input:**
- Chromosome Length (CL)
- Document or a gene (D)
- Chromosome (EC)
- Search space (V)
- Threshold value (T)
- Fitness of gene \( f(d) \)

**Output:** Chromosome \( C \)

**Begin**

1. **Initialisation:**
   - Set \( I = 0 \).
   - Set \( EC = \emptyset \).

2. Repeat steps 2.1 to 2.2 while \( (I < CL) \)
   2.1. Select a gene \( d_i \) randomly from the search space \( V \).
   2.2. If \( (d_i \notin C \text{ AND } f(d_i) > T) \) then
      2.2.1. \( EC = EC \cup d_i \)  //Add document \( d_i \) to \( EC \)

2.3. End if

3. End While

4. Copy \( EC \) to \( C \).

**End**
5.2.2.2. Selection

After the creation of initial population, the IIARGA works iteratively by applying GA operators of selection, crossover and mutation. Selection operator is used for selecting chromosomes from the population. The selected chromosomes, called parents are chosen for reproduction or crossover based on their fitness. Through this operator, selection pressure is applied on the population of solutions with the aim to pick more promising solutions to form next generation.

IIARGA uses binary tournament selection with replacement for selecting the parents, because thresholding and tournament selection make a nice combination to be used in GA (Haupt and Haupt, 2004). Moreover, the performance of roulette wheel and binary tournament selections is compared empirically on IIARGA (Appendix D). The binary tournament selection has shown to perform well than roulette wheel selection (Zhong et al., 2005; Anbumani and Nedunchezian, 2010). The binary tournament selection selects individuals based on their fitness value. Two chromosomes are selected randomly from the population. The chromosome having higher fitness value between the two selected chromosomes is selected as first parent P1. The same process is followed for selecting second parent P2. The parent selection process can be depicted as follows:

Given Population:

\[ G = \{c_1, c_2, \ldots c_N\} \]  (5.5)

where \( c_i \) denotes a chromosome, \( I = \{1,2,\ldots,N\} \)

The fitness of chromosome \( c_k \) is denoted by \( F(c_k) \). Two chromosomes are selected randomly, say \( c_i \) and \( c_j \). Then the parent \( P_k = c_k \), such that:

\[ c_k = \max(F(c_i), F(c_j)) \]  (5.6)

The basic idea of this strategy is to select the individual with the higher fitness value from two randomly chosen individuals in the population into the next generation, thus covering wider range of values. In the tournament selection, there is no arithmetical computation based on the fitness value, but only comparison between individuals by fitness value (Zhong et al., 2005). The time complexity of tournament selection is low i.e. \( O(n) \), as compared to other selection techniques (Anbumani and Nedunchezian, 2010) and therefore the fastest.
Another technique used is elitism, where the fittest chromosome from previous generation is copied over to the next generation (Billhardt et al., 2002; Cummins and O’Riordan, 2006a; Snasel et al., 2009). This is done to retain the best building block over the generations. It enhances the performance of GA by keeping the best individual from the previous generation alive throughout successive generations, till a better solution is found. It assures that best individual from the current generation is inherited to the next generation.

Say $G_i$ and $G_{i+1}$ be two population of two successive generations, where $G_i = \{c_{1i}, c_{2i}, \ldots, c_{Ni}\}$ and $c_{ki}$ is the chromosome $k$ in population $i$. The fitness of chromosome $c_{ki}$ is denoted by $F(c_{ki})$. Then chromosome

$$c_{ji} \in G_{i+1}, \text{ if } F(c_{ji}) = \max \{F(c_{j1}), F(c_{j2}), \ldots, F(c_{jN})\}$$

(5.10)

Algorithm 5.3 describes the process of Binary tournament selection and Algorithm 5.4 illustrates the elitism strategy.

---

### Algorithm 5.3: Binary Tournament Selection with Replacement

**Input:** Population of chromosomes ($G$)
Chromosome Length($n$)
$G$: Population. $G=\{c_1, c_2, \ldots, c_n\}$
Fitness of chromosome $i$ ($F(c_i)$)

**Output:** Two selected solutions for crossover.

**Begin**

1. **Initialisation:**
   
   Set $i$ to 1.

2. Repeat Steps 2.1 to 2.3 while $i<=2$.
   
   2.1. Generate a chromosome $x$ randomly, such that $c_1 \leq x \leq c_n$.
   
   2.2. Generate a chromosome $y$ randomly, such that $c_1 \leq y \leq c_n$.
   
   2.3. If ($F(x) > F(y)$) Then
       
       2.3.1. Select $x$ as Parent $Pi$.
       
       else
       
       2.3.2. Select $y$ as Parent $Pi$.
   
   2.4. End if

3. End While

**End**
Algorithm 5.4: Elitism

Begin
Input: Current Population (C_P)  
Temporary Population (T_P)
1. Sort C_P in descending order of fitness function.
2. Sort T_P in descending order of fitness function.
3. If F(C_{n-1}(T_P)) < F(C_0(C_P)) then
   3.1.1. Replace C_{n-1}(T_P) with C_0(C_P).
4. end if
End

5.2.2.3. Crossover

Crossover operator modifies the selected parents from a population to produce new offsprings by exchanging one or more of their subparts for the next generation. The idea is to generate new offsprings which are healthier than their parents to result in a better performing generation.

There are various types of crossover techniques which have been discussed in chapter 3, along with their advantages and limitations. We use two-point crossover technique to perform crossover operation in IIARGA algorithm. The two-point crossover is applied with a check for uniqueness of genes within a chromosome.

In this crossover technique, two crossover points are chosen randomly from the parent chromosomes. The values between the chosen points are exchanged, and then a careful procedure is followed to eliminate any repeated document-ids from the solution, to produce two new offsprings. It is ensured that all genes are found exactly once in each chromosome and no gene is duplicated. Figure 5.5 depicts an example of two-point crossover with unique genes criteria and Algorithm 5.5 details two-point crossover in IIARGA.

P1 and P2 are two chromosomes selected for crossover. CP1(=3) and CP2 (=7) are two cross-points selected randomly. After exchanging the genes between the two selected points, there are chances that the chromosomes become invalid. In
In this case, Chromosome 1 contains the value 3 and 24 twice, so these values get replaced by corresponding genes from chromosome 2, maintaining the uniqueness feature. To repair the two chromosomes this exchange is applied outside the crossing region. At the end, the repaired chromosomes are produced as offsprings.

![Figure 5.5: Example of two-point crossover in IIARGA](image)

**Algorithm 5.5: Two-Point Crossover in IIARGA**

**Input:** Randomly Selected solutions P1 and P2 from the population.

Length of P1 and P2: (L)

Crossover points: (x1,x2).

**Output:** New solutions P1’ and P2’.

**Begin**

1. Select two random numbers x1 and x2, such that (0<x1, x2<L and x1<x2).
2. For i= x1 to x2
   2.1. Exchange the genes of two parents such that P1’i =P2i and P2’i = P1i.
3. End for
   
   //Check for Uniqueness
4. For j= (0 to x1-1) , (x2 +1 to L-1)
   4.1. Check for duplicate genes in each chromosome P1' and P2'.
   4.2. If (duplicacy found)
       4.2.1. Swap the gene with its map in other chromosome.
   4.3. End if
5. End for

**End**
5.2.2.4. Mutation

Mutation operator is applied in IIARGA to introduce random change in chromosome structure; the existing gene(s) in a chromosome are selected randomly and are replaced by new gene(s) from the search space based on a criterion.

It helps in introducing new genetic material in the population according to some probability and avoids convergence to local optima. Mutation offers a slight amount of random search, and helps guarantee that no point in the search space has a zero probability of being examined. Higher mutations rates induce a more thorough exploration of the search space, thus is suitable in information retrieval.

For IIARGA, we use the technique of single mutation (Beasely et al., 1993; Simon, 2009; Mashagba et al., 2011) in which a single randomly selected gene in each chromosome is replaced by another gene in the search space based on the mutation probability. The old gene is replaced with the new gene only if the fitness of new gene is same as or higher than the older gene (Radwan et al., 2006; Klabbankoh and Pinngern, 2000) and it does not already exist in that chromosome. This guarantees the performance of new chromosome to be same as older chromosome or better than it. Figure 5.6 depicts an example of one-point mutation and Algorithm 5.6 details mutation in IIARGA.

![Figure 5.6: Example of mutation in IIARGA](image-url)
Algorithm 5.6: Single-point Mutation in IIARGA algorithm

**Input:** Selected solution R.
L: Length of chromosome R.
m: position at which gene may be replaced
f(x): fitness of gene x.

**Output:** New solution R'.

**Begin**
1. Select a random position m such that (0<m<L).
2. Select a random gene x from search space.
3. If \( f(x) \geq f(d_m) \) then
   3.1.1. Replace \( d_m \) with x.
4. End If
**End**

5.2.2.5. Crossover and Mutation Rate

The probability at which crossover is applied during the formation of next generation is called crossover rate. The probability of applying mutation at the time of creation of next generation is known as mutation rate. Krömer et al. (2010) have proved experimentally that large values of crossover probability and mutation probability helps in achieving better fitness values in information retrieval optimization process. They performed query evolution for every combination of crossover probability and mutation probability to find out the best combination of these probability values, and observed that the best fitness values were being attained in optimization runs with large mutation probabilities. Cecchini et al. (2008) also proved experimentally that higher mutation rates induce a more thorough exploration of the search space. Also, the review of literature showed that in IR domain the crossover rate generally ranges between 0.5 to 1 (Radwan et al., 2006, Aly, 2007; Husek et al., 2005; Snášel et al., 2009; Simon, 2009; Boughanem et al., 2002; Maitah et al., 2013; Nassar et al., 2013), and most of them have set crossover rate as 0.8. It was also noted that mutation rate ranges from 0.01 to 0.7 (Nassar et al., 2013; Al-Dallal, 2012). Thus, we are use probability of crossover \( PC = 0.8 \) and probability of mutation \( PM = 0.7 \) for the rest of experiments presented in this thesis.
5.2.2.6. Fitness Function

The performance of each encoded individual is assessed using an objective or a fitness function which assigns a fitness value to each chromosome. Fitness value represents the quality or merit of each chromosome or document as a measure of its relevance to the user query. The fitness function plays an important role in performing selection, crossover and mutation to decide about the chromosomes to be carried over to the next generation.

The fitness function can either be a standard measure (Aly, 2007; Nassar et al., 2011) or can be a customized function formulated using a set of parameters or factors best suited to that model (Radwan et al., 2006; Al-Dallal and Abdulwahab, 2009; Cummins and O’Riordan, 2006a). The standard measures are general in nature, while customised fitness function can be adapted according to the nature and needs of the problem at hand.

A fitness function comprises of a set of factors: local and global factors. Measures derived from knowledge of the document are regarded as local factors like document size, term frequency, frequency of the highest occurring term in a document, etc. Measures derived from the knowledge of the collection are regarded as global measures like document collection size, number of documents in which a term is found, cumulative frequency of a term in the document set, etc. It is generally a combination of local and global factors. This is because a term being considered in context of a document may have a high frequency not only in that document but also occurs too often in the entire collection. This will scale down the relevance of such term because of its high frequency globally. Similarly, the term having high frequency locally, but low document frequency globally is considered important as it is representative of the document content as well as has high discriminating power. Thus, both local as well as global factors play an important role in determining the role of term in the retrieval system.

Cummins and O’riordan (2005, 2006a, 2006b) used genetic programming to evolve a set of local and global weighting schemes for vector space model. They also proved the significance of \( cf/df \) measure over \( idf \) empirically. They authors used
document length as normalization factor for the log of term frequency measure. The square-root and log function were applied on the term frequency, to derive the local weighting formula given as equation 5.8.

\[ lw = \sqrt{\frac{1+\log{(rtf)}}{n}} \]  

The GP system evolved seven global weighting schemes. Most of the evolved schemes consisted of $\frac{cf}{df}$ as one of the factors. This factor measures the density of the term in the collection. The evolved global-term weighting schemes evaluated the distance between the ranked lists produced from these schemes. The best global weighting scheme which is robust and promoted useful features of the data set is given as equation 5.9.

\[ gw = \sqrt{\frac{cf^3N}{df^4}} \]  

The standard measures used for Vector space models are similarity measures of cosine, Dice, Jaccard, and Inner Product. These are not considered for calculating the fitness as VSM has not been used for indexing of the terms in IIARGA. The weights of the terms are taken into account as it is an important component and thus a key factor of the fitness functions. As there is no standard fitness measure being used for Inverted Indexed model in general, a need for developing a customised fitness measure was considered. The customised fitness function developed for IIARGA is called Document Scoring Formula (DSF), which combines the global and local term weights. The fitness value of each gene or document will contribute in fitness of chromosome.

The fitness of a gene or document $d_j$, $j=\{1, 2, \ldots, N\}$ is given as:

\[ f(d_j) = \frac{\sum_{i=1}^{M} y_i}{M} \times \sum_{i=1}^{M} w_{ij} \]  

The fitness function that evaluates the document consists of two components. The first component is a proportion or fraction of the query terms that exist in the document. This is because the documents containing more of query terms have a
The Improved Information Access and Retrieval Through Genetic Algorithm

higher chance of being related to the query. The second component of Equation 5.10 is the summation of weights of only those terms of the query that is present in the document. The weight of \( i \)th term in document \( j \) is denoted as \( w_{ij} \) (\( i=1,2,\ldots,M \)). It is calculated as follows:

\[
w_{ij} = \frac{1 + \text{log}(tf_{ij})}{\sqrt{L_j}} \times \sqrt{\frac{(cf_i) \times N}{df_i^2}} \times \frac{cf_i}{df_i} \times q_i \tag{5.11}
\]

The frequency of \( i \)th term in the query is given as \( q_i \). The details of the variables used in the formula are explained in Table 5.1. The fitness scores of each document that is a part of a chromosome is summed together to obtain the fitness value of a chromosome. Thus the fitness function used to calculate the fitness of each chromosome \( c_k \) is as:

\[
F(c_k) = \sum_{k=1}^{CL} f(d_k) \tag{5.12}
\]

where CL represents the length of chromosome \( c_k \).

Table 5.1: Formula terms and their description

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>( i )</td>
<td>( i )th term in a document</td>
<td>Local</td>
</tr>
<tr>
<td>( j )</td>
<td>( j )th document in the search space</td>
<td>Local</td>
</tr>
<tr>
<td>( y_i )</td>
<td>represents the existence of ( i )th query term in the document</td>
<td>Local</td>
</tr>
<tr>
<td>( M )</td>
<td>Total number of terms in a query ( Q )</td>
<td>Global</td>
</tr>
<tr>
<td>( q_i )</td>
<td>Frequency of term ( i ) in user query ( Q ).</td>
<td>Global</td>
</tr>
<tr>
<td>( w_{ij} )</td>
<td>weight of ( i )th query term in document ( j )</td>
<td>Local</td>
</tr>
<tr>
<td>( f(d_j) )</td>
<td>Fitness value of document ( j )</td>
<td>----</td>
</tr>
<tr>
<td>( tf_{ij} )</td>
<td>Frequency of term ( i ) in document ( j )</td>
<td>Local</td>
</tr>
<tr>
<td>( L_j )</td>
<td>Document length</td>
<td>Local</td>
</tr>
<tr>
<td>( cf_i )</td>
<td>Cumulative frequency of term ( i ) over the document collection</td>
<td>Global</td>
</tr>
<tr>
<td>( df_i )</td>
<td>Document frequency of term ( i ) or number of documents containing term ( i )</td>
<td>Global</td>
</tr>
<tr>
<td>( N )</td>
<td>Total Number of documents or collection size</td>
<td>Global</td>
</tr>
<tr>
<td>( q_i )</td>
<td>Frequency of ( i )th query term in the user query</td>
<td>Global</td>
</tr>
<tr>
<td>( CL )</td>
<td>Chromosome length</td>
<td>----</td>
</tr>
<tr>
<td>( F(c_k) )</td>
<td>Fitness value of chromosome ( k )</td>
<td>----</td>
</tr>
</tbody>
</table>
5.2.2.7. Termination Criteria

The evolutionary phase of IIARGA goes on building new generations until one of the two stopping conditions is fulfilled. Either the GA has run for 50 generations or the optimal phenotype does not improve successively up to ten percent of maximum number of generations.

The first condition of maximum number of generations is set to a higher value, as such the GA starts converging on an average by generation number 35-40. The second condition is set because the ultimate aim is to get the fittest chromosome as an outcome of GA process. The fitness strategy terminates the evolution stream if its maximum fitness does not change after a given number of generations. We have evaluated the iterative process up to five consecutive generations for improvement of the chromosomes, which is a fairly reasonable number. As the chromosome does not improve for a number of successive generations, there is less likelihood that there would be further improvement, thus GA is assumed as reaching the optimal solution. GA unit of IIARGA will stop in any of the two cases, whichever happens earlier.

5.2.3. Ranking Phase

Ranking phase ranks the document according to the relevance of the user query. IR system ranks the documents according to the similarities between document and the query. If a document has got high similarity, then that document is closer to the query. After processing the query effectively, the top most documents are displayed to the user in the order of their relevance to the query.

The ranking phase of IIARGA applies reordering to the phenotype, with the highest fitness value, obtained after the evolutionary phase is over. It involves arranging the documents in descending order of their fitness values. This will bring the fitter genes or documents to the left of the chromosome and weaker or irrelevant documents towards right edge of the chromosome, resulting in ranking the relevant documents at higher ranks. The reason of reordering is to find a gene ordering with better evolutionary potential (Beasley et al., 1993b). The reordering has been
applied at the end of GA module, because applying reordering each time the offspring is created, delays the Genetic Algorithm process.

5.3. GA Parameter Setup

Experiments were conducted to decide upon the values of Genetic Algorithm parameters like chromosome length, population size and maximum number of generations of Genetic Algorithm for the environment settings of the IIARGA.

Selecting appropriate chromosome length and population size

A set of 20 queries were selected randomly from the list of queries, such that each query selected had a distinct number of relevant documents. The implemented program was run on these set of 20 queries for 10 independent runs involving different combinations of values for chromosome length and population size.

The number of relevant documents for each query formulated for XRR collection and passed to IIARGA ranges from 2 to 41. Each chromosome represents a possible solution which is a set of possible documents relevant to the user query. As the chromosome consists of a set of the genes where each gene is a document, the chromosome length should be selected such that it covers all the possible solutions to the user query. Thus the probable values for chromosome length were taken as CS = \{50, 75, 100, 125\}.

The population may also affect the result of possible solution for a given problem. The set of potential values for population size was taken as PS = \{50, 75, 100\}. The experiment was executed for each of these combinations of chromosome length and population size for 10 independent runs, the results of which are then averaged in terms of convergence speed, average time taken, mean average precision and recall. For each query the last generation number is recorded and then averaged over the queries. Table 5.1 gives a brief summary of the various parameters analysed. The results of these experiments are discussed in the following paragraphs:

At each chromosome length, the population size was changed to 50, 75 and 100. It was seen that as the chromosome length increased, the average generations at which GA converged decreased. For each chromosome length value, the convergence
rate at different population sizes is averaged. The Genetic Algorithm took least time
to converge at chromosome length of 125, irrespective of size of population. The
average convergence at chromosome length 125 (for all population sizes) was 39.1
generations. Also, the mean average precision and recall value improved as
chromosome length increased. The decrease in average convergence rate was
highest, 10.5% at the chromosome length of 125. With the increase in chromosome
length, the average execution time taken per query increased, but the difference in
increase in that time from chromosome length 100 to chromosome length 125 was
less as compared to that when chromosome length increased from 50 to 75 and 75 to
100. This increase is 13.8% while earlier it was 50% and 31%. Thus, we choose
**chromosome length to be 125 for IIARGA.** Table 5.2 shows the percentage
changes with increase in chromosome length. Figure 5.7 shows the chart for the
change in average generations taken for convergence of the algorithm.

**Table 5.2: % Changes in parameters with varying chromosome length**

<table>
<thead>
<tr>
<th>Chromosome Length</th>
<th>Average Generation</th>
<th>% Decrease</th>
<th>Average Execution Time</th>
<th>% Increase</th>
<th>Mean Average Precision</th>
<th>Recall Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>46.52</td>
<td>--</td>
<td>0.381</td>
<td>--</td>
<td>0.6224</td>
<td>0.8504</td>
</tr>
<tr>
<td>75</td>
<td>45.27</td>
<td>2.7%</td>
<td>0.572</td>
<td>50.1%</td>
<td>0.6398</td>
<td>0.9062</td>
</tr>
<tr>
<td>100</td>
<td>43.68</td>
<td>3.5%</td>
<td>0.749</td>
<td>30.9%</td>
<td>0.6502</td>
<td>0.9445</td>
</tr>
<tr>
<td>125</td>
<td>39.11</td>
<td>10.5%</td>
<td>0.853</td>
<td>13.8%</td>
<td>0.649</td>
<td>0.9705</td>
</tr>
</tbody>
</table>

For deciding upon population size, it was analysed that at chromosome
length 50, the results obtained in terms of MAP and recall is higher at population
size 100 than that at population size 50 and 75. As the chromosome length increased
from 50 to 75, the results at population size 75 improved more in terms of recall and
average generations taken to converge when seen as a trade-off between average
execution time. Also, it was found that there was not much of a difference in mean
average precision and recall at population size 75 and that at population size 100 for
various chromosome length values, but the average convergence rate decrease at
population size 75 when chromosome length was increased from 100 to 125, the
decrease was seen to be the most that is 12% as compared to the population size 50
and 100, where the decrease percentage is 10.9% and 8.6%. If population size was
taken as 100, the average execution time per query increased 44.8% from the
average execution time per query at population size 75. Also from Table 5.2, it was found that the difference in values of MAP and recall was negligible at population size 75 and population size 100, when chromosome length was 125. Thus we chose suitable population size as 75 for IIARGA.

![Average Convergence rate](image)

**Figure 5.7: Average convergence rate at different chromosome lengths**

Table 5.3 shows the percentage decrease in convergence rate of Genetic Algorithm for various population sizes. Figure 5.8 shows the average of minimum number of generations taken by the algorithm to converge at different chromosome lengths and population sizes.

<table>
<thead>
<tr>
<th>Pop. Size (PS)</th>
<th>CS=50</th>
<th>CS=75</th>
<th>% Difference</th>
<th>CS=100</th>
<th>% Difference</th>
<th>CS=125</th>
<th>% Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>45.6</td>
<td>41.7</td>
<td>8.6%</td>
<td>41.5</td>
<td>0.4%</td>
<td>37.0</td>
<td>10.9%</td>
</tr>
<tr>
<td>75</td>
<td>46.7</td>
<td>45.5</td>
<td>2.5%</td>
<td>44.0</td>
<td>3.4%</td>
<td>38.7</td>
<td>12.0%</td>
</tr>
<tr>
<td>100</td>
<td>47.2</td>
<td>48.6</td>
<td>2.8%</td>
<td>45.5</td>
<td>6.3%</td>
<td>41.6</td>
<td>8.6%</td>
</tr>
</tbody>
</table>
The appropriate size of population was fixed at 75 and chromosome length was taken as 125 for IIARGA. The second experiment done on this data set is to decide upon the suitable number of evolutions for which the GA module should run.

![Average Generations Taken](image)

**Figure 5.8: Average convergence rate at different chromosome lengths and population sizes**

**Deciding appropriate number of evolutions or maximum number of generations**

Once the chromosome length and population size was fixed, the GA unit was executed for 10 independent runs on the same 20 queries at evolutions \( EV = \{50, 75, 100\} \). The average number of generations taken at each run is further averaged to find out average convergence rate at that evolution. Table 5.4 shows the average number of generations and average execution time per query taken at these evolutions.

The results showed that there was no difference in the average precision and minimal difference in recall value at evolution 50 and the same at evolution 75 or at evolution 100. The only difference is in average number of generations taken to converge by the algorithm at the three evolutions, and the average time taken per query which will increase. For evolution size 50, the GA converges at 38.73 generations and for evolution size 75, average convergence speed is 42.88 generations,
while average execution time per query is 0.793 second and 0.831 seconds respectively. It was found that % increase in average number of generations and average execution time per query from evolution 50 to 75 was 10.7% and 4.8% respectively; and from evolutions 75 to 100 was 15.7% and 28.4%. So, the maximum number of generations or evolutions for IIARGA is fixed at 50 generations.

Table 5.4: % Change in parameters for different evolutions

<table>
<thead>
<tr>
<th>Evolution Size</th>
<th>Avg. generation</th>
<th>% Increase</th>
<th>Avg. Execution Time</th>
<th>% increase</th>
<th>Recall</th>
<th>% Difference</th>
<th>MAP</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>38.73</td>
<td>--</td>
<td>0.793</td>
<td>--</td>
<td>0.9681</td>
<td>--</td>
<td>0.6539</td>
<td>--</td>
</tr>
<tr>
<td>75</td>
<td>42.88</td>
<td>10.7%</td>
<td>0.831</td>
<td>4.8%</td>
<td>0.9658</td>
<td>-0.2%</td>
<td>0.6542</td>
<td>0.0%</td>
</tr>
<tr>
<td>100</td>
<td>49.62</td>
<td>15.7%</td>
<td>1.068</td>
<td>28.4%</td>
<td>0.9702</td>
<td>0.4%</td>
<td>0.6552</td>
<td>0.1%</td>
</tr>
</tbody>
</table>

The values of other operators like crossover probability and mutation rate were discussed and decided in the previous chapter. Table 5.5 illustrates the values assumed for various GA parameters for IIARGA. The experiments done on other Genetic Algorithm parameters are shown in Appendix D.

Table 5.5: Value of GA Parameters for IIARGA

<table>
<thead>
<tr>
<th>GA Parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chromosome length</td>
<td>125</td>
</tr>
<tr>
<td>Population size</td>
<td>75</td>
</tr>
<tr>
<td>Maximum number of generations</td>
<td>50</td>
</tr>
<tr>
<td>Crossover rate</td>
<td>0.8</td>
</tr>
<tr>
<td>Elitism</td>
<td>1</td>
</tr>
<tr>
<td>Mutation rate</td>
<td>0.7</td>
</tr>
</tbody>
</table>

5.4. Summary

This chapter describes the IIARGA design framework. It consists of Preprocessing phase, Evolutionary phase and Ranking Phase. IIARGA uses Genetic Algorithm to produce results of good quality and ranks them in descending order of their relevance. The indexes are created using inverted indexed mapping after the stopwords have been removed and the keywords have been stemmed.
The initial population is created by selecting the documents randomly from the search space. These documents are added to the chromosome only if they do not exist prior in that chromosome and the fitness of a document is above a decided threshold value. The selection technique being used is binary tournament with replacement, because of its good results in GA. The crossover technique being used is two-point crossover provided that the genes are exchanged only if the uniqueness feature of chromosome is maintained. The mutation technique used is single point mutation, with the condition that the gene in the population can be replaced only by a gene which is fitter than the one to be replaced.

A customised fitness function is developed to assess the fitness of a gene in a chromosome. This Document Scoring Formula (DSF) takes into consideration the proportion of query terms found in a document along with the term's weight. The term's weight is composed of local and global factors. The global factor takes the density of terms in the collection as one of the factors.

When the Genetic Algorithm module terminates, reordering is applied on the optimal chromosome to get the ranked list of documents, which is displayed to the user.

This chapter also describes the parameter settings done for the Genetic Algorithm unit of IIARGA. It includes chromosome length, population size, crossover rate, mutation rate, and the stopping criteria. The best combination of parameters values of chromosome length, population size and number of evolutions are decided after empirical investigation. The performance of selection operator, crossover rate and mutation rate is compared with other variants to test the GA module of IIARGA. It is found that the chosen values give better results.