

ENHANCING RECALL AND PRECISION OF WEB
SEARCH USING GENETIC ALGORITHM

A thesis submitted for the degree of Doctor of Philosophy

By

Ammar Sami Al-Dallal

School of Information Systems Computing and
Mathematics, Brunel University

August 2012

ABSTRACT

Due to rapid growth of the number of Web pages, web users encounter two main problems, namely: many of the retrieved documents are not related to the user query which is called low precision, and many of relevant documents have not been retrieved yet which is called low recall. Information Retrieval (IR) is an essential and useful technique for Web search; thus, different approaches and techniques are developed. Because of its parallel mechanism with high-dimensional space, Genetic Algorithm (GA) has been adopted to solve many of optimization problems where IR is one of them.

This thesis proposes searching model which is based on GA to retrieve HTML documents. This model is called IR Using GA or IRUGA. It is composed of two main units. The first unit is the document indexing unit to index the HTML documents. The second unit is the GA mechanism which applies selection, crossover, and mutation operators to produce the final result, while specially designed fitness function is applied to evaluate the documents.

The performance of IRUGA is investigated using the speed of convergence of the retrieval process, precision at rank N, recall at rank N, and precision at recall N. In addition, the proposed fitness function is compared experimentally with Okapi-BM25 function and Bayesian inference network model function. Moreover, IRUGA is compared with traditional IR using the same fitness function to examine the performance in terms of time required by each technique to retrieve the documents. The new techniques developed for document representation, the GA operators and the fitness function managed to achieves an improvement over 90% for the recall and precision measures. And the relevance of the retrieved document is much higher than that retrieved by the other models. Moreover, a massive comparison of techniques applied to GA operators is performed by highlighting the strengths and weaknesses of each existing technique of GA operators.

Overall, IRUGA is a promising technique in Web search domain that provides a high quality search results in terms of recall and precision.

Acknowledgment

I would like to express my appreciation and sincere gratitude to everyone who helped me on this thesis including the following:

I must thank my parents who had a major role in the initiation of this thesis and had provided me with the worm continues moral support. I would like to thank my family; represented by my wife, sons and daughters for their support and patience, in spite of their interruption to follow their studies and help them in their projects.

Thanks must go to my supervisor Dr. Rasha Shaker, who had a larger role in understanding the nature of the material and how to conduct the experiments and analyze the results in addition to the guidance in the style and level of writing this thesis. Thanks to my supervisor Dr. Ramzi El-Haddadeh for his big role in the guidance and advice in organizing this thesis and the addition of the necessary proposals, including ideas commensurate with the components of this thesis. I thank Prof. Shawqi Al-Dallal for reviewing the thesis and providing beneficial comments. Finally, I would like to thank the database administrator from International Turnkey System Company for helping me in setting the programming environment and allows my system to run smoothly.

Contents

Error! Bookmark not defined.

List of Figures

Figure 1-1: The layout of thesis	19
Figure 2-1: The components of general IR system	20
Figure 2-2: GA process showing the role of fitness function highlighted by bold diamonds	46
Figure 2-3: Example of one-point crossover	60
Figure 2-4: Example of two-point crossover	61
Figure 2-5: Example of uniform crossover	62
Figure 2-6: Example of inversion crossover	63
Figure 2-7: Example of combining reordering crossover and 2-point crossover	64
Figure 2-8: Example of random mutation	69
Figure 3-1: The units of IRUGA	75
Figure 3-2: Sample of HTML document	77
Figure 3-3: Word node	83
Figure 3-4: Document node	83
Figure 3-5: Data structure constructed by the inverted index showing how word node is linked to a list of document nodes	84
Figure 3-6: The parser engine showing the input tables and the output tables	87
Figure 3-7: The flowchart of creating the inverted index	88
Figure 3-8: The flow of the GA process	91
Figure 3-9: Overview of GA process showing the effect of fitness functions	91
Figure 3-10: Illustration of the hybrid crossover process	97
Figure 3-11: Illustration of the applied mutation in IRUGA	99
Figure 3-12: first document retrieved by query: Digital systems design	112
Figure 3-13: example of low relevant document for the query: Digital system design"	113
Figure 3-14: Probability of success for population size = 50 and chromosome length = 50.	117
Figure 3-15: Probability of success for population size = 75 and chromosome length = 50.	117
Figure 3-16: Probability of success for population size = 100 and chromosome length = 50	118
Figure 3-17: Probability of success for population size = 125 and chromosome length = 50	118
Figure 3-18: Number of relevant documents per each query number	119
Figure 3-19: Probability of success for population size =125 and chromosome length cl = 50	119
Figure 3-20: Probability of success for population size =125 and chromosome length cl = 75	120
Figure 3-21: Probability of success for population size =125 and chromosome length cl = 100	120
Figure 3-22: Probability of success for population size =125 and chromosome length cl = 125	120
Figure 4-1: Comparison of P@N for different selection techniques	140
Figure 4-2: Comparison of R@N for different selection techniques	141
Figure 4-3: Comparison of P@R for different selection techniques	142

Figure 4-4: Comparison of P@N for different parent selection techniques	144
Figure 4-5: Comparison of R@N for different parent selection techniques.....	145
Figure 4-6: Comparison of P@R for Different Parent Selection Techniques	146
Figure 4-7: Comparison of P@N between hybrid crossover and 2-point crossover technique.....	149
Figure 4-8: Comparison of R@N between hybrid crossover and 2-point crossover technique.....	150
Figure 4-9: Comparison of P@R between hybrid crossover and 2-point crossover technique.....	151
Figure 4-10: Comparison of P@N between hybrid crossover and 2-Offspring crossover techniques.	153
Figure 4-11: Comparison of R@N between hybrid crossover and 2-Offspring crossover techniques.	154
Figure 4-12: Comparison of P@R between hybrid crossover and 2-Offspring crossover techniques..	155
Figure 4-13: Comparison of P@N between hybrid crossover and the non-ordered crossover techniques.	156
Figure 4-14: Comparison of R@N between hybrid crossover and the non-ordered crossover techniques	157
Figure 4-15: Comparison of P@R between hybrid crossover and the non-ordered crossover techniques	158
Figure 4-16: Comparison of P@N for Different Crossover Techniques for queries having more than 10 relevant documents.	160
Figure 4-17: Comparison of P@N for Different Fitness Functions	162
Figure 4-18: Comparison of R@N for Different Fitness Functions	163
Figure 4-19: Comparison of P@R for Different Fitness Functions.....	164
Figure 4-20: Comparison of P@N for Different Mutation Rates	166
Figure 4-21: Comparison of R@N for different mutation rates.	167
Figure 4-22: Comparison of P@R for different mutation rates	168
Figure 5-1: The TPBTIR layout.....	177
Figure 5-2: The first document retrieved using the TPEF function of TPBTIR.....	183
Figure 5-3: the first document retrieved using OKAPI-BM25 function.....	183
Figure 5-4: The first document retrieved using Bayesian Inference Network function.	184
Figure 5-5: The precision improvement of TPEF over other functions.....	186
Figure 5-6: The recall improvement of TPEF over other functions.....	187
Figure 5-7: The precision- recall improvement of TPEF over other functions.....	188
Figure 5-8: The first doc retrieved by IRUGA for query 7 of table 7-2.	195
5-9: The first doc retrieved by TF-IDF formula for query 7 of table 7-2	197
7-1: Two samples of the first two retrieved documents using query:" digital systems design"	238

List of Tables

1-1: The difference between Commercial Searching Engines and Academic search engines	10
Table 2-1: Summary of indexing model used in IR systems	28
Table 2-2: Summary of creating initial generation methods in GA systems	43
Table 2-3: Parent Selection techniques used in GA and their advantages and disadvantages	57
Table 2-4: Summary of crossover techniques used in GA	66
Table 2-5: Summary of mutation methods used in GA	71
Table 3-1: The weight assigned to HTML tags used in the inverted index	78
Table 3-2: Terms of formulas 3.1 and 3.2 showing their description, domain and type.	104
Table 3-3: Terms of formulas 3.4 showing their description, domain and type	106
Table 3-4: Probability of success for different values of the population size.....	118
Table 3-5: The probability of success of IRUGA for population size = 125 and different chromosome length.....	121
Table 3-6: Parameter setting of IRUGA	123
Table 4-1: Categories of documents used to test IRUGA.....	131
Table 4-2: Statistics of the test collection used in IRUGA	132
Table 4-3: Data sets used in similar IR researches.	132
Table 4-4: The parameters used in the measurement of document retrieval (Horng and Yeh, 2000).....	135
Table 4-5: The average convergence of each technique.....	138
Table 4-6: The P@N enhancement percentage for different selection techniques.....	140
Table 4-7: The R@ N enhancement percentage for different selection techniques	141
Table 4-8: The P@R enhancement percentage for different selection techniques	142
Table 4-9: The P@N enhancement percentage for different parent selection techniques	145
Table 4-10: The R@N enhancement percentage for different selection techniques	145
Table 4-11: The P@R enhancement percentage of for Different Parent Selection Techniques	147
Table 4-12: The P@N enhancement percentage of hybrid crossover over the 2-point crossover techniques	149
Table 4-13: The R@N enhancement percentage of hybrid crossover over the 2-point crossover techniques	150
Table 4-14: The P@R enhancement percentage of hybrid crossover over the 2-point crossover techniques	151
Table 4-15: The P@N enhancement percentage of hybrid crossover over the 2- Offspring crossover techniques	153
Table 4-16: The R@N enhancement percentage of hybrid crossover over the 2- Offspring crossover techniques	154
Table 4-17: The P@R enhancement percentage of hybrid crossover over the 2- Offspring crossover techniques.	155
Table 4-18: The P@N enhancement percentage of hybrid crossover over the non-ordered crossover techniques	156
Table 4-19: The R@N enhancement percentage of hybrid crossover over the non-ordered crossover techniques.	157

Table 4-20: The P@R enhancement percentage of hybrid crossover over the non-ordered crossover techniques.	158
Table 4-21: The P@N improvement of hybrid crossover over other crossover techniques for queries having more than 10 relevant documents.	161
Table 4-22: List of fitness functions	161
Table 4-23: The P@N enhancement percentage of the term proximity fitness function over other fitness functions.....	162
Table 4-24: The R@N enhancement percentage of the term proximity fitness function over other fitness functions.....	163
Table 4-25: The P@R enhancement percentage of the term proximity fitness function over other fitness functions.....	165
Table 4-26: The P@N enhancement percentage of MUTE70 over other mutation rates	166
Table 4-27: The R@N enhancement percentage of MUTE70 over other mutation rates	168
Table 4-28: The P@R enhancement percentage of MUTE70 over other mutation rates.	169
Table 4-29: Summary of the techniques, measures and percentage of improvement. The percentage of improvement beside each technique represents the average comparison of this technique with the one in bold above it.	172
Table 5-1: The performance of TPBTIR	178
Table 5-2: Comparison between IRUGA and TPBTIR in terms of recall and precision measures.....	180
Table 5-3: Comparison between IRUGA and TPBTIR in terms of query processed time	181
Table 5-4: Comparison of the ranks and factors of the three fitness functions for the first document retrieved by each fitness function.	185
Table 5-5: Percentage of precision improvement of TPDF over other functions	187
Table 5-6: Percentage of recall improvement of TPDF over other functions	188
Table 5-7: Percentage of precision- recall improvement of TPDF over other functions	189
Table 5-8: Summary of TPDTIR and IRUGA performance	189
5-9: Factors used to calculate TF-IDF for document 2.....	195
Table 5-10: Summary of the performance of the IR techniques discussed	203
Table 7-1: description of the terminals used in weighting and fitness functions.....	228
Table 7-2: Term weighting formulas used in GA systems	229
Table 7-3: Fitness functions used in GA systems to measure document relativity to user query.	231
Table 7-4: The retrieved documents by SRS and PRS techniques using query: “digital systems design”.....	236
Table 7-5: The retrieved documents by PS100 and PS75 techniques using query: “digital systems design”	239
Table 7-6: The retrieved documents by HC and C2O techniques using query: “ <i>multimedia systems</i> ”.....	241
7-7: The retrieved documents by HC and C2O techniques using query: “multimedia systems”	244
7-8: The retrieved documents by HC and NOC techniques using query: “multimedia systems”	247

Table 7-9: : example of the results obtained by TPFf and OKAPI-BM25 fitness functions using the query "database management systems" 250

Table 7-10: example of the results obtained by TPFf and bayesian inference network model fitness functions using the query "database management systems" 252

DECLARATION

Research from this thesis that has been published is presented below.

Journal publications

1. **Al-Dallal, A.** and Abdulwahab, R. (2012) GA and IR: study the effectiveness of the developed fitness function on IR, *International Journal of Artificial Life Research (IJALR)*, 3(2), Apr-Jun
2. **Al-Dallal, A.**, Abdulwahab, R. S., and El-Haddadeh, R.: IR with and without GA: Study the Effectiveness of the Developed Fitness Function on the Two Suggested Approaches, *International Journal of Applied Metaheuristic Computing (IJAMC)*. (Accepted to be published in IJAMC). (2012).

Conference papers:

1. **Al-Dallal, A** (2012), Enhancing Recall and Precision in Web Search using Advanced Fitness Function, *CD-ROM/Online Proceedings of the European, Mediterranean and Middle Eastern Conference on Information Systems (EMCIS)*. 6-7 June, Munich, Germany. (pp 29-41).
2. **Al-Dallal, A** and Abdulwahab, R. (2011) GA-based System for Achieving High Recall and Precision in Information Retrieval, *Proceedings of the 1^{0th} Biennial International Conference on Artificial Evolution (EA-2011)*, October 2011, Angers, France.
3. **Al-Dallal, A.** and Abdulwahab, R. (2011) Achieving High Recall and Precision with HTLM Documents: An Innovation Approach in Information Retrieval, *Proceedings of the World Congress on Engineering 2011 (WCE 2011)*, July 2011, London, U.K.
4. **Al-Dallal, A.** and Abdulwahab, R. (2009) Genetic Algorithm Based to Improve HTML Document Retrieval, *Proceedings of Second International Conference on Developments in eSystems Engineering (DeSE'09)*, Dec. 2009, Abu Dhabi, UAE.

-
5. **Al-Dallal, A.** and Abdulwahab, R. (2009) Genetic Algorithm in Web Search using inverted index representation, *Proceedings of the 5th IEEE GCC Conference and Exhibition*, March 2009, Kuwait City, Kuwait.

Chapter One: Introduction

1.1 Overview

The World-Wide Web provides users with access to an abundance of information. Users query particular information from the Web using web search engines, and these web search engines apply the information retrieval (IR) techniques to produce the needed information. Information Retrieval is primarily devoted to extracting relevant information in response to user query. The increasing amount of information on the web raises new and challenging problems for information retrieval which is denoted as web search problem.

These problems can be summarized in two areas, namely, the inaccuracy of the retrieved information from the web (Bhatia and Khalid , 2007) within an acceptable retrieval time (Kobayashi and Takeda, 2000). In order to keep the focus, only the first problem will be addressed in this thesis. Moreover, the information that can be retrieved using information retrieval system (IRS) has multiple forms including text documents, sound files, images, videos, etc. In order to focus the work in this thesis, only the text documents will be considered for investigation and improvement.

The first web search problem has been investigated by many researchers attempting to develop approaches that are capable of providing search results that satisfy user query, examples are: (Liu, 2006; Marghny and Ali, 2005; Picarougne et al, 2002a; Kim and Zhang, 2000; Fan et al, 2004; Kushchu, 2005; Karthik, Marikkannan, and Kannan, 2008; Snasel, Moravec, and Pokorny, 2005; Tian et al 2006; Bhatia and Khalid, 2007; Kobayashi and Takeda, 2000; Haveliwala et al, 2002; Ashraf, Ozyer, and Alhajj, 2008; Yan et al, 2009; Xu, Deli, and Yu, 2009; Saini, Sharma, and Gupta, 2011). Often, these results are evaluated using precision and recall perspectives. For precision, it measures the percentage of relevant retrieved documents to the total retrieved documents, while recall measures the percentage of relevant retrieved documents to the total relevant

documents in search space.

In spite of several enhancements achieved in such approaches, still web users encounter two major challenges when trying to retrieve useful information (LEE, 2007; Bhatia and Khalid, 2007; Haveliwala et al 2002; Pathak, Gordon and Fan, 2000); namely; low precision and low recall. Low precision is due to the irrelevance of many of the search results where many of the highly ranked retrieved documents are not related to the user query (Picarougne et al 2002a). On the other hand, the second challenge is the low recall, which is due to the inability to index all the web documents available on the Web and related to the user query, bearing in mind that the aim of the searching engine is to retrieve all relevant documents based on the user query (high recall), and not to retrieve any irrelevant document (high precision).

1.2 Information Retrieval Techniques

Recently, the IR challenge has gained certain importance as it aims at satisfying user requirements by providing the most relevant set of documents in response to his query. Thus, researchers have contributed to enormous paradigms and technique to solve this issue. These techniques have been classified into four categories; probabilistic IR, knowledge based IR, IR based on machine learning techniques, and traditional IR (Pathak et al 2000; Chen, 1995; Dong, 2008). In probabilistic IR (Fuhr, 1992), the probabilistic retrieval is based on estimating a probability of relevance of a document to the user query. Typically, relevance feedback from a few documents is used to establish the probability of relevance for other documents in the collection (Fuhr and Buckley, 1991; Gordon, 1988). The second category of IR is knowledge-based IR (Chen and Dhar, 1991). This approach focuses on modelling two areas. The first area tries to model the knowledge of an expert retriever in terms of the expert's domain knowledge. The second modelling area is the user of the system. This modelling area is similar to the way used by the librarian to develop the client profile. Although knowledge based approaches might be effective in certain domains, it may not be applicable in all domains (Chen and Dhar, 1991). The third category is learning-systems based IR (Pathak, Gordon and Fan, 2000). This approach is based on algorithmic extraction of knowledge or on identifying patterns in the data. There are three broad areas within this approach: symbolic learning,

neural networks, and evolution based algorithms (Pathak, Gordon and Fan, 2000; Dong, Hussain, and Chang, 2008). In the symbolic learning approach (Quinlan, 1986, 1993), knowledge discovery is done typically through inductive learning by creating a hierarchical arrangement of concepts and producing IF-THEN type production rules. Neural networks are connectionist learning algorithms that typically simulate the way the human brain learns and remembers knowledge. In these algorithms knowledge is captured and remembered in terms of the weights on synapses, the interconnections of the neurons, and the thresholds on logic units (Chen, 1995; Azcarraga et al, 2012; Guezouli and Kadache, 2012). Evolutionary algorithms are based on the principles of natural selection (Marghny and Ali, 2005). These algorithms can be further divided into: Genetic Algorithms (GA), evolutionary strategies, and evolutionary programming. GA is based on genetic operators of selection, crossover, and mutation (Holland, 1975), while evolutionary programming utilizes changes at the level of species (Fan, Fox, Pathak, and Wu, 2004), and the evolutionary strategies exploit changes at an individual behavioural level (Fan et al, 2004). The fourth category is the traditional IR (TIR) (Dong, 2008). This category is based on the semantic search engines and methods which are derived from the traditional index-term-based information retrieval models. These models are further classified into three main categories, namely, *Boolean* models, *Algebraic* models and *Probabilistic* models. However, the general issue in these TIR models is that they are computationally costly, where all documents in the search space need to be evaluated against all of the indexed documents (Dong, 2008).

By investigating the above techniques, one can deduce the following: The first category which is probabilistic IR requires parsing the whole document set upon receiving the user query in order to estimate the probability of relevance of a document to the user. This process requires the user to wait for a longer time before getting the final results. Hence, it is impractical from a time perspective and from processing resources. Moreover, this technique applies one factor only to evaluate the relevance which is based on relevance feedback from a few documents to establish the probability of relevance for other documents in the collection. The second category, which is knowledge-based IR, and both symbolic learning and neural networks, require preparing a model/example of the data or applying the technique to a sample set before generalizing it to the whole

document set and producing the final results. This process consumes additional resources, making it impractical in the domain of web search (Chen, 1995). According to (Dong et al, 2008), the extensive computation cost is a common drawback for most of the Traditional IR paradigms as it has to be applied to a vast number of documents in the data set. Moreover, it is unreasonable to solve web-search problems using traditional IR techniques. This is because in such techniques, all documents must be processed and evaluated to produce the results related to the user query. However, considering the huge number of web documents available in the search space makes this solution impractical from a time processing point of view, since the user is going to wait for very long time before he gets the results of his query. On the other hand, if the IR system is going to process a limited number of documents in order to be fast enough, then it could not retrieve all relevant documents and may process and retrieve unrelated documents ending with very low recall and very low precision.

By looking at the methods of evaluate the documents, it is seen that most popular method is based on an evaluation formulas which combines set of factors. One can deduce that these factors are either limited to the statistical factors such as frequency of terms within the document/collection, frequency of unique terms within the document/collection, frequency of the most frequent term within the document, total number of documents referring to a particular keyword(s), total number of documents in the search space, etc (Salton and Buckley, 1988; Vrajitoru, 2000; Kim and Zhang, 2003; Radwan et al 2006; Cummins and O’Riordan, 2006; Radwan et al 2006; Aly, 2007). Or these factors are limited to the semantic factors where some researches use some HTML tags in weighting the terms and hence in evaluating the documents (Kim and Zhang, 2003). It is clear that there is a room to combine more factors and consequently to create more advanced evaluation formulas to better reflect document relevancy to the user query.

1.3 Web Search, Artificial Intelligence and Software Engineering

Emphasis on the application of artificial intelligence (AI) to IR has been increased in recent years as an alternative approach to traditional IR systems with the aim of solving IR problems. That is because extraction of requested information requires searching for it

among tremendous collection of documents. Hence, Search is inherent to the problems and methods of AI as AI problems are intrinsically complex. Efforts to solve problems with computers which humans can routinely solve by employing innate cognitive abilities, pattern recognition, perception and experience, invariably must turn to considerations of search. All search methods essentially fall into one of two categories, either: exhaustive (blind) methods or: heuristic (informed) methods (Bini et al, 2009). Since the domain of the proposed model is a collection of vast number of documents and requires an efficient technique, the first method becomes impractical to be adopted. Hence, the best searching method that suits the proposed question is the second one which is the heuristic method. Through knowledge, information, rules, insights, analogies, and simplification in addition to a host of other techniques, heuristic search aims to reduce the number of objects examined. However, heuristics do not guarantee the achievement of a solution, although good heuristics should facilitate this. On the other hand, heuristic search represents a practical strategy increasing the effectiveness of complex problem solving (Bini et al, 2009). It leads to a solution along the most probable path, omitting the least promising ones (Amarel, 1968), and it enables avoiding the examination of dead ends while using already gathered data (Lenat, 1983).

In the domain of web search, heuristic methods can be applied to decide which documents to examine, instead of examining all documents in the search space, and also deciding that certain documents should be discarded, or pruned, from the selection process (Kopec and Marsland, 1984).

AI as pointed by (Rech and Althoff, 2004), has two main strands, one is the scientific strand and the other is the engineering strand. The scientific strand deals with the cognitive science which requires the interference of the software engineering (SE) in order to implement such system efficiently. In fact, there is a strong overlap between SE and the engineering strand of AI. The important part of AI is the knowledge base systems (KBS). Richter (2004) defines three different levels to describe KBS: the cognitive layer (human-oriented, rational, and informal), the representation layer (formal, logical), and the implementation layer (machine-oriented, data structures and programs) which represent the area where SE is involved. Its main concern is the efficient and effective development of high qualitative and mostly very large software systems (Rech and

Althoff, 2004).

1.4 Genetic Algorithm

Most studies argue that IR can be seen as a standard optimization problem (Marghny and Ali, 2005,), where it has search space S represented by the set of documents, a set of possible solutions S^+ (the possible documents related to the user query), and evaluation function f to evaluate the relevance of each of these possible documents related to the user query. Finally, a search engine tries to output documents that maximize f . The optimal solution is a document or set of documents that have the maximum score returned by the function f . It is found that such an optimization problem can be solved efficiently using Genetic Algorithm (Klabbankoh and Pinnern, 2008; Kim and Zhang, 2003; Petridis et al, 1998; Kazarlis et al, 2001). In addition, GA requires less processing resources compared with the approaches mentioned in the previous section, since there is no need to apply the technique to the training set before finding the optimum solution. Moreover, there is no need also to evaluate all documents in the search space in order to find the optimum solution. Therefore, it has been adopted by many researchers because of its simplicity, flexibility, robustness, its capability as a powerful search mechanism, and the ability to provide parallel solutions simultaneously (Aickelin, 1999; Kim and Zhang, 2003; Sivanandam and Deepa, 2008); it can also be employed to make several important contributions to the field of IR. Moreover, genetic search algorithms enable intelligent and efficient internet searches and they are especially useful when the search space is relatively large, as in the web (Milutinovic, Cvetkovic and Mirkovic, November 2000). GA uses the principles of selection and evolution to produce several simultaneous solutions to a given problem (Sivanandam and Deepa, 2008); thus, it doesn't stick to a local optimum solution, rather, as it is a heuristic technique, it can permits to find in average near optimal solution (Bini et al, 2009).

There have been a considerable number of various approaches investigating GA engines for solving the web search problem (Kosala and Blockeel, 2000; Marghny and Ali, 2005; Radwan et al 2006; Pathak, Gordon and Fan, 2000; Vrajitoru, 1997; Klabbankoh and Pinnern, 2008; Yan et al, 2009; Drias, Khennak, and Boukhedra, 2009; Xu, Deli, and Yu, 2009). In these types of approaches, GA generates an initial population which is a

group of individuals or set of documents selected randomly from the search space. The individuals in the population are then evaluated using what is called a fitness function. The fitness function is provided by the programmer and gives the web documents a score to measure their relevance to the user query. Two individuals are then selected based on their fitness: the higher the fitness, the higher the chance of being selected. These individuals are then "reproduced" by an operation called crossover to create one or two offspring, after which they are mutated randomly. This continues until a suitable solution has been found or a certain number of generations have been passed, depending on which happens first.

In the literature of using GA in IR it is found that the following techniques and areas of GA are studied. Marghny and Ali (2005) elaborate on the quality of the retrieved web documents by generating mean quality functions which utilize link quality and page quality. Radwan et al (2006) develop a fast and flexible fitness function. This fitness function depends on the difference between term weights of a given chromosome and the query vector to evaluate the retrieved web documents. Pathak et al ,(2000) apply GA to adapt matching functions that are used to match document descriptions with queries. The performance of the system is measured using recall and precision. However, these measures are based on a predefined document cut-off value, which is the number of documents the user is willing to see. In addition, this approach lags in performance, using a limited number of factors to evaluate individuals and being applied to a limited set of documents. Vrajitoru (1997) maintained the relevance judgments of the past queries in order to improve the performance of the system on the current query. However, this approach uses simple GA operators except that it tries to estimate the error rate in a reliable way by separating the information used for the evaluation from that used in the training phase. While the above mentioned research works on search space to produce a better result, Klabbankoh and Pinngern (2008) use GA to optimize the query chromosome for document retrieval, and also study the effect of the probability of crossover and mutation on recall and precision.

From the above mentioned studies of GA in IR, it has been found that there is a need to enhance recall and precision through improving the quality of retrieved documents and by displaying them at the top. Consequently, the user can be satisfied with the proper

results that appear right at the beginning of the retrieved list of document rather than scrolling down and browsing multiple pages before finding the page that he/she wants (Bedi and Chawla, 2007). Looking at the literature of using GA in IR from another perspective, it has been found that many studies including Kim and Zhang (2003), Pathak et al (2000) Aly (2007), Vrajitoru (2000), Martín-Bautista and Vila (1998) and Klabbankoh and Pinngern (2008) have utilized the classical vector space model in their studies to present their documents. Such an approach is considered to be impractical as it requires a large space, a long time to create, and a long time to process (Snasel, Moravec and Pokorny, 2005). On the other hand, Song and Park (2009) have used the Latent Semantic Index model (LSI), which is relatively better than the vector space model in the sense that it reduces the storage space and reduces the time of re-processing. However, it consumes a longer time to generate the index as it is constructed in two stages. The first stage is the building of the classical vector space, and the second stage is the construction of the semantic vector extracted from the classical vector space (Song and Park, 2009). Another point in existing GA is the techniques applied to evaluate the documents. Most of these techniques are based on statistical factors and these factors are a combination of local and global factors (Kim and Zhang, 2003; Radwan et al, 2006; Billhardt et al, 2002; Vrajitoru, 2000; Cummins and O’Riordan, 2006; Salton and Buckley, 1988). Local factors are those obtained from the document itself such as document size, term frequency, and number of unique terms within the document, while global factors are obtained from the collection, such as total number of documents, total number of terms in the collection, total number of unique numbers, etc. These types of factors make the evaluation of the document depend heavily on the collection, making the degree of relevance inaccurate. In this case, the retrieval of the document is based on the documents in the collection rather than reflecting their actual relevance, e.g. for a particular query, there is no document exactly relevant to this query, but when using these factors, the system returns the closest document in the collection to this query, although its actual relevance is very small or non-existent. Another drawback in the existing GA models is that most of their techniques are applied to plain text and not to structural documents (Pathak et al, 2000; Aly, 2007; Vrajitoru, 2000; Martín-Bautista and Vila, 1998; Klabbankoh and Pinngern, 2008), although most web documents are structured or semi-

structured. However, Kim and Zhang (2000; 2003) use GA to obtain the best HTML tag weight for terms within the HTML text documents.

1.5 Commercial Searching Engines

No one can deny the importance and popularity of the commercial searching engines (CSE) such as Google, Yahoo, MSN, etc. Even so, a lot of research is still conducted to solve the web searching problems mentioned above. Why? In fact, the behaviour of these search engines rarely considers the structure of the web document (Tian et al, 2006; Kim and Zhang, 2003); rather, it considers intensively the links pointing to the retrieved web document. As stated by Callen (2005, p. 21), the pages are ranked by Google based on the importance and the number of websites that link to the retrieved document. In general, a website will have high page ranking when it has a high number of links pointing to it (Callen, 2005, p. 21), or if it has a link from a high ranked page pointing to it (Kim and Zhang, 2003). Google uses three main techniques to evaluate the document. The first one is the PageRank. The second one is location of the information which applies the proximity concept, and finally, the visual presentation details such as the font size of words (Green, 2004). The PageRank for a web page is evaluated using the following technique:

For a web page A that has pages T_1, T_2, \dots, T_n , which are point to it, this web page has also number of links going out of it, denoted by $C(A)$, the PageRank of web page A is given as follows:

$$PR(A) = (1-d) + d (PR(T1)/C(T1) + \dots + PR(Tn)/C(Tn))$$

Where parameter d is a damping factor that can be set between 0 and 1 but usually set to 0.85. As the PageRank forms a probability distribution over web pages, the sum of all web pages' PageRanks will be one.

Even though, scientist and researchers are still developing other techniques to keep on improving and proposing more and more IR techniques. Marghny and Ali (2005) used Google results as an input to their IR model in order to further improve the quality of its results. Moreover, it is noted that the users may query the same search engine several times by rephrasing the query in order to obtain the desired results (Tian et al, 2006).

Also this is evidence of the low quality of the retrieved results if it is assumed that the user passed the correct query to the search engine. Another point worth clarifying is that the title is mentioned as “commercial”, which implies that the commercial effect plays a vital role in such types of search engine. In fact, there is a big difference between academic research applied to IR and CSE (Glover, 2007). This is due to various factors. One of them is the crawling: How much of the web is crawled? What is the update frequency? Not all web pages use HTML, etc. The second factor is indexing: what to index, dealing with polysemous and synonyms, dealing with updates, etc. The third factor is query processing: many queries would have large relevant sets, the order of query words, geographical place of the user, year of the query, etc. The last factor is that the user interface issues are extremely important for the retrieved web pages (Glover, 2007). These factors make the CSE largely disconnected from the academic search engine. In addition, the CSE is characterized by some biasing features (Glover, 2007). For Example, ranking the results cannot assume independence, content of a web page is not sufficient to imply meaning, results must consider maliciousness, the web is large or infinite, and the goal of the CSE is to make money. The differences between CSE and Academic search engines are summarized in table 1-1. Needless to say that the proposed model cannot replace the commercial engines; however, the proposed techniques can be included or added to the techniques applied by these CSE.

1-1: The difference between Commercial Searching Engines and Academic search engines

Commercial Searching Engines	Academic Searching Engines
Goal is to make money	Goal is to retrieve relevant documents
UI is extremely important	Retrieving relevant documents is extremely important
Real-time/fast expectation	Concerned with quality of retrieved document regardless of the time
Content of web page not sufficient to imply meaning	Content of web page is sufficient to imply meaning
Result ranking cannot assume independence	Result ranking assume independence most of the time
Must consider maliciousness	Maliciousness is not considered at all
No quality control on pages (quality varies)	The quality of pages is controlled
Web is large (practically infinite)	Data set is finite
Millions of heterogeneous users	Limited users

1.6 Problem Statement and Research Motivation

Every internet user wishes to have satisfactory results when using any web search engine. Satisfaction is in the sense that all the retrieved results are relevant and all relevant documents are retrieved; in other words, the web user is satisfied when the information retrieval system retrieves *all* and *only* the relevant documents within a reasonable response time. In spite of several enhancements having been achieved in such search techniques, still web users encounter two major problems when trying to retrieve useful information (Kosala and Blockeel, 2000; Pathak, Gordon and Fan, 2000; Haveliwala, 2002; Cho and Richards, 2004; Yeh et al, 2007; LEE, 2007; Bhatia and Khalid, 2007; Bedi and Chawla, 2007). The first problem is low precision. It is due to the irrelevance of many of the retrieved results, where many of the highly ranked retrieved documents are not relevant to the user query (Picarougne, 2002). The second problem is low recall, which is due to the inability to index all the relevant web documents available on the Web.

The ultimate objective of any information retrieval system (IRS) is to retrieve *only* the relevant documents. In this case, the precision will be equal to one. Another ultimate objective is to retrieve *all* the relevant documents which yields in a recall of one. Since, in real situation, it is not practical to display all relevant results to the user query, researchers consider an asymptotic solution, where the results are evaluated at the first N retrieved document; given that N is multiples of 10 and $0 \leq N \leq 100$. The common values of N for the precision are 10, 20, 30, ..., 100. Some authors use the average of these 11 points (Cutler et al, 1999; Kim and Zhang, 2000) while others use the first two values, which are precision at 10 and 20 (Kim and Zhang, 2003). Considering this measure, it is shown from the literature that the maximum results achieved for the average 11-points measure is 0.255 (Cutler et al, 1999). When using $N < 3$, or precision at first 10 and 20 retrieved documents only, then the best result obtained is the average of these 2 points which is 0.545 (Kim and Zhang, 200).

The second measure that is used by researchers is the recall at first N retrieved documents, where N is similar to that of the precision measure. This measure gives accurate results for the proposed measure (that is recall at N and $N \leq 100$) when the total

relevant documents for a given query is not more than 100 documents. Otherwise, this evaluation becomes relative, where the results are compared with other techniques or with other queries for same technique undertaken. When retrieving all the relevant documents within first 100 positions, then this measure will return a score of one for recall at 100 retrieved documents. However, authors use the same concept of the average of 11-points which is mentioned above. In this case the maximum score achieved in this domain is 0.319 (Cho and Richards, 2004).

To overcome the limitation of the number of retrieved documents compared with the number of all relevant documents, especially for the web, researchers use the concept of precision-recall measure. This measure combines both precision and recall into one measure. In fact, this is a more popular measure that is used in evaluating the information retrieval systems than the previous two (Horng and Yeh, 2000, Desjardins, Godin, and Proulx, 2005, Aly, 2007, Yeh et al, 2007, Kim and Zhang, 2000). This evaluation technique measures the precision at each 10% of the total relevant documents retrieved. In another words, when retrieving 10% of the total relevant documents, what is the percentage of these 10% within the total retrieved documents? As an example, suppose that 5 documents form 10% of the 50 relevant documents. By the time the system has retrieve these first 5 relevant documents, it has also retrieved an additional 3 irrelevant documents. In this case, the precision-recall = $5 / (5+3) = 0.625$. So the precision-recall at 10% = 0.625. Obviously, it measures how many impurities (irrelevant documents) exist within the displayed results. The ultimate score of this measure is also one and it is achieved when retrieving only the relevant documents and displaying them at the top ranked position.

Similar to the 11 points used for other measures, researchers use the same concept of the average 11 points for the precision-recall measure, where the domain of the relevant documents is divided into 10 slots and it evaluates the percentage of the relevant documents within each slot. Then the average of these slots is computed. By looking at the scores achieved using this measure it is found that the best score reaches 0.7003 (Horng and Yeh, 2000). In addition, it is found that the score achieved for these measures using current approaches is still far away from user expectation where his/her expectation that need to be accomplished is to have an information retrieval system that is able to

achieve a score of one or very close to one when evaluated by these measures. Therefore, this thesis aims at enhancing the recall and precision of the web search to achieve an average 11-point precision higher than 0.35, an average 11-point recall higher than 0.9 and an average 11-point precision-recall higher than 0.9 in order to meet the user desire when accessing the web search engine.

This aim will be achieved through three steps. The first one will focus on the document representation. The second step is to modify the existing GA operators such as initial generation creation, parent selection, crossover and mutation. The third direction is to develop a fitness function that is able to distinguish the relevant documents from irrelevant by giving a high score for the relevant documents.

1.7 Research Aim and Objectives

As Web search becomes a vital area for all Web users, there is a need to have a robust search mechanism that is able to display all relevant documents as the top ranked results. This research aims at providing such a mechanism which enhances the precision by displaying the relevant documents at top rank, and enhances the recall by retrieving as many relevant documents as possible from the search space. The maximum average precision of the existing approaches reaches 0.255, while the maximum average recall is 0.319. When using precision-recall measure, this score reaches 0.7003. This research aims to produce higher scores for the same measures by applying the GA model to HTML documents using the enhanced inverted index. Based on the motivation for the work mentioned earlier, the objectives to be achieved by this research are:

1. To develop an enhanced inverted index that takes an $O(n)$ time to construct and retrieve the needed data, requires small storage space and retrieve the data in $O(n)$ time, where n is the size of the indexed terms. The design of this index is based on analyzing the different existing models of document representations to benefit from their advantages and overcome their weaknesses.
2. To explore the existing techniques of genetic algorithms which are used in information retrieval and identify their advantages and weaknesses to produce higher performance operators and develop a new technique for the crossover

operator which enhances the retrieval results in terms of precision and recall.

3. To propose new evaluation function to evaluate the documents retrieved that compete with the existing document evaluation functions after identifying their advantages and disadvantages. This function need to combine local factors, statistic factors and semantic factors, so the evaluation is done independently of other documents in the search space. These factors expected to enhance the recall and precision by at least 10% compared to the benchmark fitness functions (OKAPI-BM25 and Bayesian inference network model).
4. To propose new crossover technique that doubles the quality of chromosomes within each generation.
5. To develop traditional IR (TIR) model based on the proposed evaluation function to compare its performance with IRUGA from one side and with other GA-based IR techniques from other side in terms of time, recall and precision. The time can be measured using the actual time in seconds, and in case of GA the time will be compared in terms of maximum number of generations produced by the model.

1.8 Research Methodology

Implementing an IR-system Using GA (IRUGA) model requires robust data structure that is capable of supporting and maintaining a large amount of data and at the same time provides fast access and retrieval of the requested information. However, it was not mentioned explicitly in the literature what kind of such tools are used, but as per (Application development: PL/SQL, Java or C++?, 2002) it is found that the Oracle database is much more suitable and provides a powerful tool that is featured by consistency and faster access to the data needed than C++. Therefore, the Oracle 10g database and PL/SQL programming language are going to be used for generating the inverted index and modelling IRUGA since Oracle supports the large number of documents to be indexed and is very straightforward in storing the data required, retrieving the terms and documents needed, sorting the genes within the chromosome, and manipulating the data. Also it is very easy and simple to program compared with C++ which requires additional effort to manipulate the character string and find the

minimum distance between terms whereas it can be done in Oracle using a simple SQL statement. C++ also requires an extra code for sorting while it can be done in SQL easily by just adding *order by* closure to the SQL statement. Moreover, several difficulties are encountered while treating large numbers of documents using C++ from a memory management point of view.

This is from the environment and tools point of view. From the structure point of view, IRUGA will be composed of two main units, namely: indexing unit and GA unit. In the indexing unit, the inverted index is created to transform the documents into a structure that can be accessed by IRUGA easily and efficiently. The GA unit is composed of several operators which are the initial selection process, the parent selection operator, the innovative hybrid crossover operator, and the mutation operator. These operators act to form consecutive generations. Each generation is composed of a set of chromosomes. Each chromosome represents a possible solution. These chromosomes are evaluated using specially developed fitness function. The best chromosome of the last generation represents the results of the user query. These results are encoded in the form of the index of documents relevant to the user query.

Results obtained by IRUGA will be analyzed by studying four measures. The first measure is the speed of convergence, where the number of generations required to produce the final result is going to be compared with other similar techniques applied to the IR domain. The second measure is the precision at top N , where N is the top retrieved document and $N \in \{0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$. The average precision at these points starting from 0 to 100 forms 11 points; hence, it is called 11-point average precision. This measure is commonly used to evaluate the IR systems developed (Cutler, Deng et al, 1999; Kim et al, 2000). The third measure is the recall at top N . This measure evaluates how many relevant documents exist in the window of size N , where N is the same as above. The last measure which is common in this domain is precision at recall percentage, noted as P@R measure (Radwan, 2006; Aly, 2007; Desjardins, 2005; Horng and Yeh, 2000; Kim and Zhang, 2000). This measure calculates the percentage of relevant retrieved document to each 10% of the total number of relevant documents. In other words, it is the percentage of relevant documents when retrieving 10%, 20%,... 100% of total relevant documents.

1.9 Thesis Outline

The remainder of this thesis is organized in the following way:

- The second chapter starts by highlighting the main components of the general IR system. Then it describes three types of documents before reviewing and assessing the document representation models. The common models of implementing IR systems are presented in order to show where the proposed GA fit among them. An investigation of the current techniques applied to implement the GA operators is performed. It shows the advantages and the drawbacks of these techniques. Finally, a discussion of different types of fitness functions is provided.
- Chapter three introduces the design of IRUGA. It is composed of two main components: the document representation and the GA mechanism. At the beginning it explains the features of the HTML document as a document type adopted for IRUGA. Next, in the same section, the reason behind the selection of specific HTML tags and their weights is highlighted. The description of the first unit of IRUGA, namely, the inverted index unit is explained in full details. The advantages of this indexing model are also discussed. The next section explains the main unit involved in IRUGA, namely, the GA unit. This chapter proposes a set of operators that can be applied by the GA unit in such a way that IRUGA at the end will be capable of producing the high quality results expected. Finally, the chapter closes by introducing two fitness functions that are used to evaluate the documents retrieved by the GA unit.
- Chapter four describes the environment configuration of the machine used to run IRUGA. The reason behind choosing PL/SQL and oracle 10g for implementing this approach will be discussed. The experimental work for choosing IRUGA's parameters is also presented in this chapter. Next, several experiments are conducted to setup IRUGA's parameters. These parameters include population size, chromosome length, crossover probability, mutation probability, and the terminating criteria. Additionally, an experiment is conducted to test how the performance of IRUGA is influenced by these parameters. Finally, this chapter

- defines a range of experiments that are used to evaluate and compare each technique proposed for IRUGA units against the existing technique.
- Chapter five presents the evaluation process of IRUGA. It starts by describing the document set which is written in HTML format and forms the search space of IRUGA. Next it describes the queries used to examine the IRUGA performance. The evaluation measures used to assess IRUGA are also explained in this chapter. These measures are: precision at rank N , recall at rank N , precision at recall and the convergence speed. This chapter employs the experiments as explained in Chapter 4 and evaluates the proposed techniques of implementing the GA operators which are described in Chapter 3. The analysis of each experiment is performed and illustrated by tables and diagrams. Finally, a table that summarizes all the experimental results performed throughout this chapter is included at the end.
 - Chapter six compares IRUGA, which is a GA-based IR model, with traditional IR (TIR) systems. This chapter starts by providing a description of the TIR structure and its units. Next, a TIR system is built based on the term proximity evaluation function (TPEF) forming: Term proximity-based TIR (TPBTIR). Then, the performance of this system is compared with IRUGA in terms of the processing time in one experiment, and in terms of the precision and recall measures in the second experiment. In order to illustrate the novelty of the proposed term proximity function, an experiment is conducted to compare the quality of the documents retrieved by this function and documents retrieved by other two well known evaluation functions using the TIR system. These functions are the OKAPI-BM25 function and the Bayesian inference network function. The last experiment, tests the performance of both IRUGA and TIR using these three evaluation functions. This experiment shows that if the processing time is not a concern, then TIR is better only for the TPEF functions. This chapter concludes that IRUGA is the best approach for a large set of documents such as the web, while TPBTIR is the best for a small collection of documents.
 - Chapter seven shows how IRUGA and TPBTIR managed to enhance the

precision and recall measures. This is done by analyzing the results obtained in Chapters 5 and 6, and comparing them with the performance of existing GA-based IR models described in Chapter 2 in terms of P@N, R@N, and P@R measures. The results obtained by IRUGA reflect the novelty of this model and the achievements aimed in this study since it is more suitable than TPBTIR for the web search domain.

- Chapter eight concludes this thesis. It summarizes the design of the proposed IRUGA and TPBTIR models, followed by the contributions and the conclusions drawn from this work. The limitations of this work are listed before opening the door to the possible directions of future work for extending the proposed model to web search space.

The layout of thesis is presented in Figure 1-1.

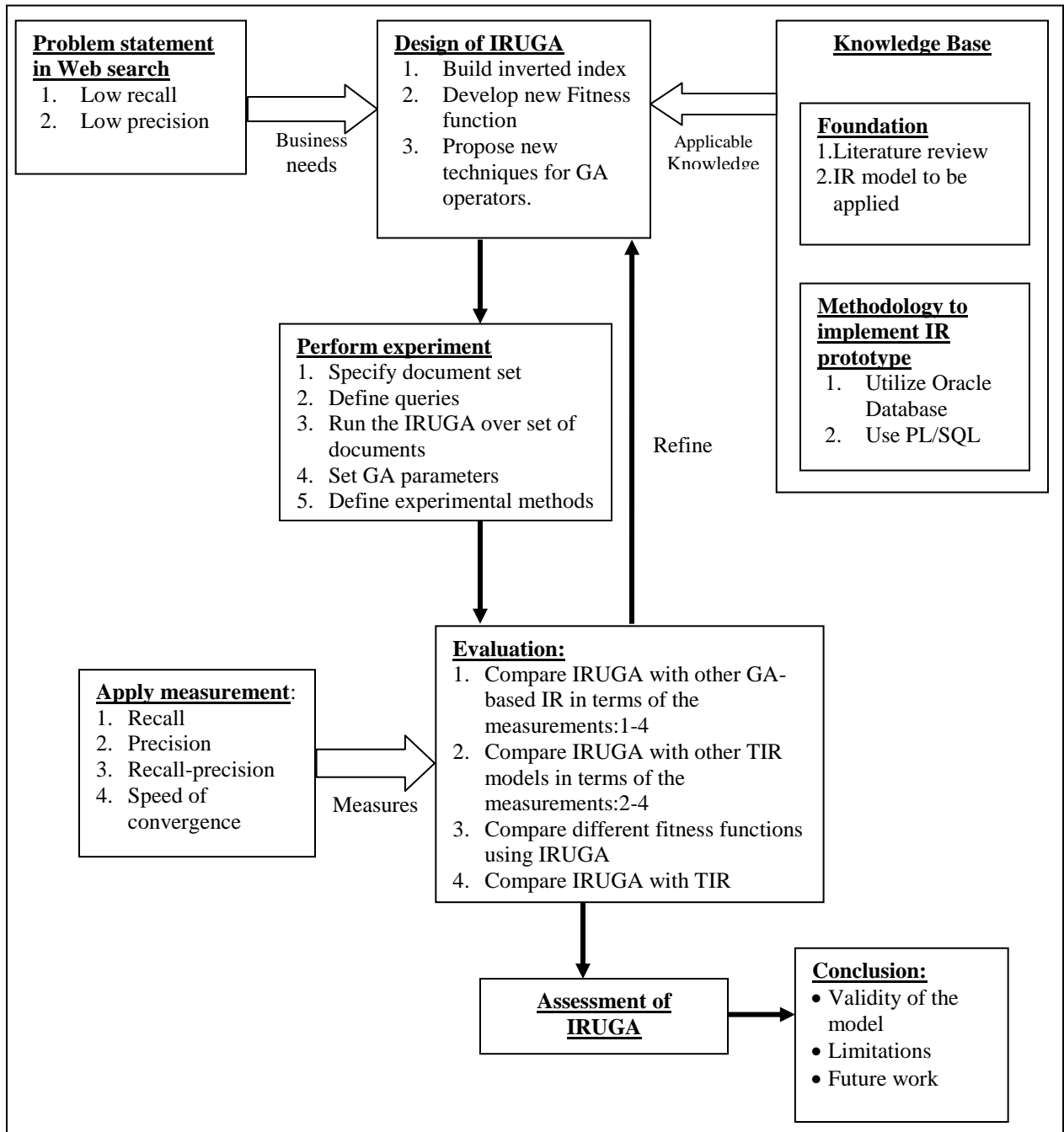


Figure 1-1: The layout of thesis

Chapter Two: Literature Review

2.1 Overview

Emphasis on the application of artificial intelligence (AI) to information retrieval (IR) has been increased in recent years as an alternative approach to traditional IR systems with the aim of solving IR problems. One of the AI areas is evolutionary computation (EC), which is based on models of natural selection. A classical and very important technique in EC is the genetic algorithm (GA). The GA is biologically inspired and has many mechanisms derived from natural evolution. Because of its parallel mechanism with high-dimensional space, GA has been used to solve many scientific and engineering problems. This in turn began to encourage researchers to use this algorithm in the field of IR. Moreover, GA played an important role in providing suitable information for the user's needs. IR systems in general are composed of four components. These components are: collection of indexed documents and user query as an input, the retrieval engine as a processor which has the ability to evaluate documents based on user query, and a set of ranked retrieved documents as an output. These components are illustrated in Figure 2-1.

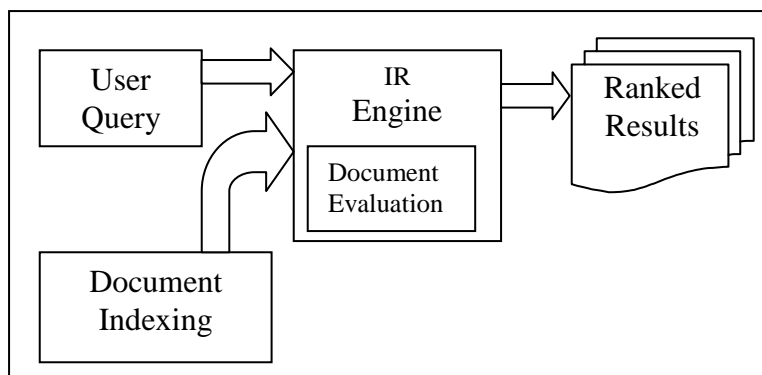


Figure 2-1: The components of general IR system

For any IR system to be applied to a set of documents, these documents need to be represented in such a way that the IR system can easily retrieve the relevant document as a response to a user query. This representation is done through a process called document

indexing. This chapter is going to review different techniques used to index the documents in order to identify their advantages and weaknesses, then it will consider GA systems as an approach applied to the IR domain by looking into each operator of GA and indentifying its drawbacks and limitations whenever they exist. Since genetic algorithms have become more popular, intensive research work has been directed to develop this domain. Therefore, the literature review is restricted to review specific areas related to GA systems that are used in information retrieval systems only.

This chapter is organized as follows: Section 2.2 starts by discussing the basic three types of documents followed by presentation of several models of document representation. Section 2.3 is about the basic information retrieval techniques which are Probabilistic IR, Knowledge Based IR, Learning Systems Based IR and traditional IR. The main work is presented in Section 2.4 where it discusses the techniques used in GA to solve the IR problem. GA is based on three operators and each one of these operators is elaborated in a separate sub-section. It starts with the initial generation creation methods. Next is the elaboration of the main three operators of GA, namely: selection, crossover and mutation. Then it emphasizes the intensive work undertaken to develop the fitness function used to evaluate the documents.

2.2 Document Representation

When talking about document representation, there is a need to take a look at the document types that form the search space of any IR system. This in turn, leads to an investigation of several document representation models that are applied as a re-processing stage for IR systems. What is meant by document representation is the process of extracting the meaningful words from a document and presenting them into a structure that facilitates the process of matching the query with the documents referencing these words. These models will be discussed in this section by highlighting their advantages and drawbacks followed by an assessment of them.

2.2.1 Types of documents

Basically, an IR system can be applied to several kinds of documents. These documents are categorized into three classes: structured, semi-structured and non-structured

documents.

When a collection of documents shares the same kind of information, it is natural to think about describing that information in the same way for all of them. For example, conference papers have almost the same structure, so it is desirable that this structure would be described in a standard way. These ideas were at the origin of the standard for specifying markup languages, the *Standard Generalized Markup Language* (SGML), as a format of exchanging documents (Gancarski and Henriques, 2003). Some SGML documents have a well defined hierarchical structure, such as titles, subtitles and headers. So they are called *structured documents*. The sections in structured documents are clearly marked with single or multiple levels headings. Structured documents can have other attributes which are necessary to create the hierarchy, such as distinctive colour, underlines, boldness, etc. (Alam et al, 2003). An example of this type of document is eXtensible Markup Language (XML). The specification of the structural elements and their hierarchical relations for a given type of documents is made through the Document Type Definition (DTD) (Gancarski and Henriques, 2003). The advantage of this type of document is that both type and location of data is known before scanning (Kofax, 2011). On the other hand, if the type of data is known but its location is not known before scanning, then this type of document is called *semi-structured* (Kofax, 2011). A famous example is the HTML documents. HTML documents are developed as an SGML application to show the documents in the Web (Gancarski and Henriques, 2003). This standard makes use of presentation marks to describe how the textual parts must be displayed by the browser. Contradictory to XML which uses Document Type Definition, HTML has limited but not a small number of types of tags, hence they are embedded in the same file.

The third type of documents is the unstructured document or *flat* document which referred to as plain text in many papers. In this type of documents, both the type and location of the information it contains are unknown prior to scanning (Kofax, 2011). Also it will not have any of the attributes mentioned in the structured document such as headers, colour, underlines and boldness (Alam et al, 2003). These types of documents usually have a title, but after that the content is not organized in any structured fashion. It is interesting to note that much research in the IR domain uses this type of document. The

reason is that many document sets in this format are served by a predefined index, a set of queries, and a set of documents relevant to these queries. Such an approach offers simplicity for developers to utilize this system.

2.2.2 Models of Document Representation

The IR system is applied to a set of documents which form the search space. These documents must be represented in a way such that matching these documents with queries is easy (Pathak, Gordon and Fan, 2000). Another consideration in document representation is that such representation should correctly reflect the author's intention (Pathak, Gordon and Fan, 2000). The primary concern in representation is how to select proper index terms (Pathak, Gordon and Fan, 2000) and which indexing model is to be implemented. Typically, representation proceeds by extracting keywords that are considered as content identifiers and organizing them into a given format (Pathak, Gordon and Fan, 2000). The basic method of web search and traditional IR system is to find documents that contain the terms in the user query. Many models such as the Boolean Model, Vector Spacing Model, Probabilistic Model, Latent Semantic Indexing Model and Inverted Index Model have been developed to represent the documents.

2.2.2.1 Boolean Model

The Boolean model is one of the oldest and simplest information retrieval models. In the Boolean model, documents and queries are represented as sets of terms. That is, each term is only considered present or absent in a document. Using vector representation of the document, the weight of the term t_i in document d_j is 1 if t_i appears in document d_j , and 0 otherwise. Given a Boolean query, the system retrieves every document that makes the query logically true. Thus, the retrieval is based on the binary decision criteria, i.e., a document is either relevant or irrelevant. There is no notion of partial match or ranking of the retrieved documents. This is one of the major disadvantages of the Boolean model, which often leads to poor retrieval results (Billhardt, Borrajo, and Maojo, 2002).

2.2.2.2 Vector Spacing Model

The vector space model (VSM) is the most commonly used model (Liu, 2006; Lopez-

Pujalte, Guerrero-Bote, and de Moya-Anegon, 2003a). In this model, a document is represented as a weight vector, in which each component weight is computed based on some variation of term frequency TF, or term frequency- inverse document frequency TF-IDF. The weight w_{ij} of term t_i in document d_j is the number of times that t_i appears in document d_j .

In this model, the documents are ranked according to their degree of relevance to the query. One way to compute the degree of relevance is to calculate the similarity of the query \mathbf{q} to each document \mathbf{d}_j . There are many similarity measures, such as the cosine similarity (Aly, 2007; Vrajitoru, 1997; Lopez-Pujalte, Guerrero-Bote, and de Moya-Anegon, 2003a), which is the cosine of the angle between the query vector \mathbf{q} and the document vector \mathbf{d}_j . Another way to assess the degree of relevance is to directly compute a relevance score for each document to the query. The Okapi method and its variations are popular techniques in this setting (Liu, 2006).

When implementing GA-based IR system using the aforementioned technique by (Aly, 2007) using VSM and cosine similarity function, the average 11-point precision-recall measure achieved ranges between 0.2969 and 0.4321. Another variant of the VCM is called context vector space (CVM) implemented by (Billhardt, Borrajo, and Maojo, 2002). CVM is a semantic indexing technique that uses co-occurrence data to estimate the *probability-based semantic meaning* of a term or its *context* in relation to other terms. This technique is tested using various mid-sized document sets. The achieved average precision varies between 0.2839 and 0.6746 reflecting instability effect of such model.

The vector spacing model has many disadvantages (Snasel, Moravec and Pokorny, 2005). One of them is that the document vectors have a big dimension (e.g. 150,000) and are quite sparse. The second is called the “curse of dimensionality”, which causes classical indexing structures, such as M-trees and A-trees, to perform in the same way or even worse than sequential scan in a higher dimension. The third is the synonyms of terms and other semantically related words that are not taken into account. In spite of its disadvantages, still VSM is used by some researchers to implement their approaches such as (Kui and Juan, 2012) who applied an improved version of TF-IDF and called it: TF-IDF-IG. The achieved average precision and recall are 86.764% and 88.264%. This technique has improved the fitness function TFIDF by introducing the optimized feature

extraction to avoid the data imbalance problem that results from magnitude of categories.

2.2.2.3 Probabilistic Model

This model tries to use the probability theory to build the search function and its operation mode. The information used to compose the search function is obtained from the distribution of the index terms throughout the collection of documents or a subset of it. This information is used to set the values of some parameters of the search function, which is composed of a set of weights associated with the index terms (Radwan et al, 2006). However, the term weight within this model depends basically on the probability of the word within the document regardless of its position in the document. Accordingly, it loses the ability to distinguish between documents and gives the same level of relevance for documents having the same probability for the queried term.

2.2.2.4 Latent Semantic Indexing Model (LSI)

The purpose of LSI is to extract a smaller number of dimensions that are more robust indicators of meaning than individual terms (Song & Park, 2009). Once a term-by-document matrix is constructed, LSI requires the singular value decomposition of this matrix to construct a semantic vector space. Singular Value Decomposition (SVD) is performed on the matrix to determine patterns in the relationships between the terms and concepts contained in the text. The SVD forms the foundation for LSI. Due to the word-choice variability, the less important dimensions corresponding to “noise” are ignored. A reduced rank approximation to the original matrix is constructed by dropping these noisy dimensions.

Although this model requires less storage space at the last stage of construction, it has two drawbacks. The first one is that once the index is created and there is a need to add new searchable documents, then the terms that were not known during the SVD phase for the original index are ignored. These terms will have no impact on the global weights and learned correlations derived from the original collection of terms (Song and Park, 2009). Another drawback is that it requires a longer time to construct as it involves an additional step to construct SVD. (Zaman and Brown, 2010) compared between three document evaluation functions on TREC-8 data set that contain 131,321 documents using LSI

model. These functions are TF-IDF, log-entropy, and row term-frequency. The size of the matrix required to construct the index is 131,321 (the number of documents) * 93,909 (the number of indexed terms). Assuming that each term required 4 bytes of storage (if the word size is 4 characters and each character needs one byte) then the total storage size required to store this index is 49GB. The best average non-interpolated precision achieved in this study is 0.0436 and it is scored by the TF-IDF term weighting scheme. These results reflect the low efficiency of this model.

2.2.2.5 Inverted Index Model

In its simplest form, the inverted index of a document collection is basically a data structure that attaches each distinctive term to a list of all documents that contain the term along with its position within the document, the frequency of its appearance in the document, and the weight of this term with respect to the document in addition to any extra data required such as the offset of the sentence that includes this term. In this model as described in (Uematsu et al, 2008, p.308), the inverted index holds word position data, as well as document ID. Word position data is a list of offsets at which the words occur in the document. Such occurrence information (i.e. document ID and word position data) for each word is expressed as a list, called an “inverted list”, and all the inverted lists taken together are referred to as the inverted index. The position data is mainly used for sophisticated phrase (i.e. order-sensitive) searches and proximity search which depend on the distance between terms. When a phrase query is submitted, the search engine accesses the inverted list of each word of the query – referred to as the keyword - to identify documents that contain those keywords. In addition, it retrieves the additional data associated with the keyword. These additional data are fed into the evaluation process such as the fitness function. Finding documents containing multiple queries is also easy as well.

2.2.2.6 Sentence-Based Inverted Index Model

This model is similar to the inverted index model described above but it indexes only the presence of each word in each sentence. The inverted list holds sentence position data instead of word-based position data (Uematsu et al, 2008). This model reduces the size of

the index by 25% compared with word-based index. However, it doesn't reflect the accurate relativity of the document to the user query since it doesn't consider the order of the terms within the sentence or the distance between the keywords. It checks only the presence of the keywords in the same sentence within a document regardless of their order of appearance. This model is applied by (Uematsu et al, 2008) to TREC-8 data set containing 528,155 documents. The precision at first 5 retrieved documents (P@5) achieved is 0.468 which is better than word-based index by 0.8%. However, the precision at the points: P10, P20, P30, P100 and P200 for the word-based index are better where they range between 2.75% and 6.2%. Thus, when considering the precision measure, word-based index comes in the first position outperforming the sentence based index.

2.2.3 Assessment of the indexing models:

Investigation of the structure of the above models reveals that the first 3 models (Boolean, vector spacing and probability models) require scanning the document database sequentially in the retrieval process to find the documents that contain the query terms. However, these methods are obviously impractical for a large collection, such as the Web (Liu, 2006), since scanning the document database sequentially takes a relatively long time for retrieval. The Latent Semantic Indexing model ends with a relatively small index compared to vector space. However, to build that index it first uses the vector space model – with its drawbacks mentioned earlier - then it builds the modified index out of the vector space. Accordingly, it takes more time and consumes more space and resources to construct. Yet there is room for a better model which utilizes very much lower space and hence causes the information retrieval to be faster. This model is implemented using data structure (called indices) from the document collection to speed up the retrieval or search (Liu, 2006). The inverted index, which has been shown to be superior to most other indexing schemes, is a popular one and perhaps it is the most important index method used in search engines as mentioned in (Liu, 2006). This indexing scheme not only allows different retrieval of documents that contain the keywords, but is also very fast to build since it requires a one-time parse for the document set and also it supports highly sophisticated queries (Uematsu et al, 2008). Thus, in retrieval, it takes a constant time to find documents that contain a query term

along with all data related to the keyword such as the total frequency within a document and within the whole set, etc. An empirical study is conducted by (Uematsu et al, 2008) shows that the sentence-based inverted index performs slightly better than the word-based inverted index in terms of retrieval time where it is faster by 0.9%. Eventhough, the word-based index is better in terms of recall and precision.

Because the word-based inverted index model is fast to build, smaller in storage space, stores word position rather than sentence position, is mainly fast to retrieve the needed data, and it produces better results in term of precision and recall, it is adopted for IRUGA instead of any other indexing model. Advantages and disadvantages of each indexing model are listed in Table 2-1.

Table 2-1: Summary of indexing model used in IR systems

Model of presenting the documents	Description	Advantages	Disadvantage
Boolean model (Radwan et al 2006)	<ul style="list-style-type: none"> Binary indexing: the term either exists or not (0 or 1). 	<ul style="list-style-type: none"> Provide an exact match for documents having the same query term only. 	<ul style="list-style-type: none"> Requires a large space to present document. Requires a long time to process a document, $O(n^2)$. Only presence or absence of term is provided. No additional term info (e.g. frequency, weight, and order within document) is provided.
Vector space model (Lopez-Pujalte, Guerrero-Bote, and de Moya-Anegon, a, 2003; Aly, 2007; Vrajitoru, 1997; Billhardt et al, 2002)	The document is viewed as a vector in n-dimensional document space, where n is the number of distinguishing terms and each term represents one dimension in the document space.	<ul style="list-style-type: none"> Documents are evaluated based on frequency of the query terms. Readymade Datasets represented as vector space are available in the web. Many techniques of evaluating documents' similarity to the query are available to be used with this indexing model 	<ul style="list-style-type: none"> Requires a large space to present document where many redundant zeros are presented in the vector. Requires a long time to process a document, $O(n^2)$. Adds extra info associated to the terms (e.g. frequency, weight, etc.) enlarges the space tremendously. Synonyms are not taken into account
Probabilistic Model (Radwan et al 2006)	Uses probability theory to build the search function and its operation mode. The information used to compose the search function	<ul style="list-style-type: none"> Term weight is evaluated based on its distribution in the data set. 	<ul style="list-style-type: none"> Evaluates terms based on term probability. Long time to construct as it requires parsing the documents to obtain term

	is obtained from the distribution of the index terms throughout the collection of documents or a subset of it.		probability before assigning the weight for each term, $O(n^2)$
Latent Semantic Indexing (Song and Park, 2009)	Once a vector space is constructed, Singular Value Decomposition (SVD) is performed on it to determine patterns in the relationships between the terms and concepts contained in the text. Based on the word-choice variability, the less important dimensions corresponding to “noise” are ignored and reduced rank approximation to the original matrix is constructed by dropping these noisy dimensions.	<ul style="list-style-type: none"> • Indexes the most significant terms only. • Term occurrence is presented by real number reflecting its local or global weight. • Presents the most important semantic information in the text using Singular Value Decomposition (SVD) • Reduces noise and other undesirable artefacts of the original space. 	<ul style="list-style-type: none"> • Requires a longer time to construct. • Requires a large space to construct (while building the index) • Cannot add extra terms to the index once created.
Word-based Inverted Index model (Liu, 2006; Uematsu et al, 2008)	It is a data structure that attaches each distinctive term to a list of all documents that contain the term along with position within the document and additional useful data.	<ul style="list-style-type: none"> • Ability to index all meaning terms. • Ability to add all needed info associated to the terms such as term frequency, term weight, etc. • Small space to store the indexed terms since only the indexed terms are stored. • Short time to construct $O(n)$ where n is total indexed terms. • Fast to access and retrieve needed terms and their related info, $O(n)$ in worst case. • Supports highly sophisticated queries 	<ul style="list-style-type: none"> • If user is not interested in the order of terms within the sentence then this will occupy a larger space than Sentence-Based Inverted Index Model
Sentence-Based Inverted Index Model (Uematsu et al, 2008)	It is a data structure that attaches each distinctive term to a list of all documents that contain the term along with a sentence within the document and additional useful data (order of words within the sentence is not maintained)	<ul style="list-style-type: none"> • Ability to index all meaning terms. • Ability to add all needed info associated to the terms such as term frequency, term weight, etc. • Small space to store the indexed terms since only the indexed terms are stored. 	<ul style="list-style-type: none"> • Exact match to user query is lost since order of words within sentence is not considered.

		<ul style="list-style-type: none"> • Short time to construct $O(n)$ where n is total indexed terms. • Fast to access and retrieve needed terms and their related info, $O(n)$ in worst case. • Supports highly sophisticated queries • Useful if order of query terms is not important 	
--	--	---	--

2.3 Information Retrieval Techniques

This section highlights various research paradigms commonly applied to IR as classified by (Pathak , Gordon and Fan, 2000; Dong et al, 2008) and where this work fits in. At a broad level, research in IR can be categorized into three categories (Chen, 1995; Dong et al, 2008): *Knowledge based* IR, IR based on *machine learning* techniques and *traditional* IR. Explanation of these categories is in the coming subsections.

2.3.1 Knowledge Based IR

This approach focuses on modelling two areas. First: it tries to model the knowledge of an expert retriever in terms of the expert's domain knowledge, that is, his or her search strategies and feedback heuristics. An example of such an approach is the Unified Medical Language System. Another area that has been modelled is the user of the system. This typically follows the way the librarian develops a client profile. Although knowledge based approaches might be effective in certain domains, it may not be applicable in all domains (Chen and Dhar, 1991).

2.3.2 Learning Systems Based IR

This approach is based on algorithmic extraction of knowledge or identifying patterns in the data by acquiring knowledge automatically from examples such as from source data or from training data sets (Chen, 1995). This is in contrast to the performance systems which acquire data from human experts. There are three broad areas within this approach: Symbolic Learning, Neural Networks, and Evolution Based algorithms. Although these techniques are derived from different origins and behaviours, all show high capability for

analyzing both qualitative, symbolic data and quantitative numeric data. Below is brief overview of each one of them.

2.3.2.1 *Symbolic Learning*

In the symbolic learning approach knowledge discovery is done typically through inductive learning. It is applied by inducing a general concept description to best describe the positive and negative examples. Another model of symbolic learning algorithms is incremental where it produces a hierarchical arrangement of concepts for describing classes of objects. The output of such a model is concept hierarchies or a set of production rules. From that, a hierarchical arrangement of concepts is created to produce IF-THEN type production rules. Examples of the positive and negative rules algorithms are the ID3 decision-making algorithm (Quinlan, 1986) and Mitchell's (1982) Version Space. ID5R (Utgoff, 1989) which is an extended form of ID3 is an example of incremental algorithms.

2.3.2.2 *Neural Networks*

Neural networks are connectionist learning algorithms that typically simulate the way the human brain learns and remembers knowledge. In these algorithms knowledge is captured and remembered in terms of the weights on synapses, the interconnections of the neurons, and the thresholds on logic units. (Azcarraga et al, 2012) applied back propagation network of neural network to generalize the relationship of the title and the content of 2000 articles by following word features. In addition to TF-IDF, these features include position of word in the sentence, paragraph, or in the entire document, and formats such as heading, and other attribute. An extraction rule algorithm is then applied to convert the back propagation networks for the two datasets into equivalent rule-sets that are more comprehensible to humans. The results are evaluated using F-measure. The achieved results of this technique ranging between 0.7836 and 0.8831 depending on the data set since the former results are from news articles and the later are from scientific journals. (Guezouli and Kadache, 2012) build an Information retrieval model based on neural networks using the neighbourhood. In this technique, and during the learning phase, for each term of a class of the output layer, the common neighbours are kept.

These neighbours represent the current term in most of the documents of the class. This technique is implemented on set of 425 documents and the achieved precision is 0.3604.

2.3.2.3 Evolutionary Algorithms

Evolutionary algorithms are based on the improvement principles of natural selection. These algorithms can be divided into: evolutionary programming EP, evolutionary strategies, and Genetic Algorithms (GAs). Evolutionary programming utilizes changes at the level of species, while evolutionary strategies are more specific and exploit changes at the individual behavioural level (Fan et al, 2004). The main feature of EP is the use of complex data structure, such as tree, link list and stack. Moreover, the structure length is not fixed, although it may be constrained to be within a limited size. In contrast to EP, GA uses simple structure to represent its elements where each individual is represented by a fixed-length bit string, like (1011011...), or by a fixed-length real number (2.3, 1.4, 3,..) (Fan et al, 2004). GAs are used to solve difficult optimization problems, while EPs are typically used to approximate complex and nonlinear functional relationships (Koza, 1992). More about GA technique will be presented in Section 2.4.

2.3.3 Traditional IR

The fourth category is traditional IR (TIR) (Dong et al, 2008). This category is based on the semantic search engines and methods which are derived from the traditional index-term-based information retrieval models. These models are further classified into three main categories, namely, *Set theoretic* models, *Algebraic* models and *Probabilistic* models. A brief description of each is in the following:

2.3.3.1 Set Theoretic Model

The *Set Theoretic algorithm* is based on set theory and Boolean algebra. A set is a collection of abstract objects where each object is a member of this set. Set theoretic models have four types. The first type is *Boolean algebra*. It is a set of logical operations between two sets such as conjunction, disjunction and complement. In the Boolean model, the appearance of the indexed term determines the weight between the term and

the document. If any conjunctive component from a query has a counterpart in a document, the document matches the query and a weight of 1 is awarded, otherwise a weight of zero is awarded. An example of applying Boolean algebra is implemented by (Yoshioka and Haraguchi, 2005). They propose a method for modifying a given Boolean query by using information from a relevant document set. This method is based on the assumption that the (pseudo-) relevant document set should satisfy the newly constructed Boolean query. As a result of this query reformulation process, some important keywords may be excluded which causes difficulties when searching for relevant documents that contain these excluded keywords. To overcome this difficulty, they propose a method that combines both the probabilistic IR model and the Boolean IR model. This system uses a modified version of the Okapi system as a probabilistic IR engine and it uses both a word index and an index of phrases comprising combinations of two adjacent words. This system is evaluated using precision-recall measure and the score achieved precision ranges between of 0.7 for R@10 and 0 with average of 0.31. This low score comes from two drawbacks of this technique. First one is the exclusion of some important keywords during the query reformulation process. The second drawback is that when expanding a query's terms using the relevant documents in the probabilistic IR model, there is a chance that documents without all the required query terms will receive a higher score than documents with these terms which reduces the retrieval performance. Generally speaking, Boolean model is useful in case of exact match is required (Lashkari, Mahdavi and Ghomi, 2009) and in this case all the retrieved documents will have similar degree of relevance which is not true from the logical point of view since many other factors need to be considered, such as the frequency of the term in the document, or the density (frequency compared to the document size), the location of the terms within the document, and other factors which will be discussed later in this chapter..

The second type of set theoretic models is the Case-Based Reasoning model. It is used to retrieve and reuse existing problems (Carthy et al, 2003). It consists of four processes: retrieve, reuse, revise and retain. The features extracted for inferred situation are given a weight depending on its certainty ranging from -1 for complete uncertainty to 1 for complete certainty. When applying Case-Based Reasoning to problem-solving (Liu et al, 2008), an association rule mining is used to discover context-based inference rules from

historical problem-solving logs. The discovered patterns identify frequent associations between context information and situation features in order to be used in inferring more situation features. By considering the inferred situation features, case-based reasoning is then employed to identify similar situations. However this study does not provide quantitative results.

The third type is the *fuzzy set* model (Dong et al, 2008). In this model, thesaurus is defined as a term-term correlation matrix. The elements of the matrix are the correlation values between two terms. In order to match the semantic similarity between documents and a query, the query expression is converted into set of conjunctive components. Then, each conjunctive component associates with a fuzzy set of documents and the union of the fuzzy sets are processed by Boolean operations. Finally, the membership value of each document in the processed fuzzy set is computed and ranked. (Alzahrani and Salim, 2009) applied the fuzzy IR on a set of 500 Arabic document and analyze their similarity compared to a set of 15 queries. This approach indexed the terms within the document by building unique term pairs. For each pair of terms, a term-to-term correlation factor that defines the extent of relevance between these two pairs is computed. Then the documents are evaluated by measuring the degree of similarity between the document under consideration and the query document. Although this model is tested on HTML document set, but the author treat the HTML as stop words. In addition, this approach constructs a term-to-term correlation matrix thesaurus which includes all unique pairs of terms extracted from the document set and the document query set. The results obtained by this approach are ranging between 0.7 and 0.73 for precision measure which it is ranging between 0.68 and 0.7 for recall measure. It is mentioned by the author that one disadvantage of this technique is that it requires long time and consumes large space.

The fourth type of the semantic search models is the *Extended Boolean* model. It uses the same concept of *Boolean algebra* model except that it uses the term-frequency inverse-document-frequency (TF-IDF) functions to measure the similarity between the document and the query (Pohl, Zobel and Moffat, 2010). For that, it is using the VSM to index the documents. (Pohl, Zobel and Moffat, 2010) used this technique for the systematic biomedical views. Instead of using recall measure to evaluate the model, they compute the average recall after retrieving collection of documents. The best result achieved is

average of 79% of relevant documents are retrieved when retrieving total of 10,000 documents. Precision of this model can be inferred if we know that the average recall is 18% at first 100 retrieved documents. This assists to conclude that the precision of this system is very low.

2.3.3.2 Algebraic Models

The second category of traditional IR is the algebraic model. This model uses algebraic formulas to find the distance between the document and the query (Kleinberg and Tomkins, 1999). This category is further divided into three models (Dong et al, 2008). This first one is the *vector space model* (VSM). In this model, the documents are indexed using the vector space model. And the relativity is measured using the common (TF-IDF) functions. This model is widely used in information retrieval studies such as (Hammo, 2009; Penev and Wong, 2010; Kui and Juan, 2012).

The generalized form of this type is called *Generalized Vector Space Model* (GSVM). It is the second type of the algebraic models and differs from the vector space model by using independent indexed terms where the set of vectors is linearly independent and forms a subspace of interest. Two vectors can be composed of smaller components which are derived from the particular collection. If the weights of association between index terms and documents are all binary, all possible patterns of term co-occurrence can be represented by a set of 2^t minterms. In the GVSM, a set of pairwise orthogonal vectors associated with a set of minterms are introduced and a set of vectors is adopted as a basis for the subspace of interest. In the GVSM, these representations can be directly translated to the space of minterm vectors. The resultant document vectors and query vectors are then used to compute the ranking using the standard cosine similarity function. The third type of the algebraic models is the *Latent Semantic indexing* (LSI) model. In LSI, a smaller number of dimensions are extracted from the vector space model to produce more robust indicators of meaning than individual terms. After constructing a term-by-document matrix, LSI requires the singular value decomposition of this matrix to construct a semantic vector space. In LSI, a term-document association matrix is decomposed into three components using singular value decomposition. The first one is the matrix derived from the term-to-term correlation matrix; the second one is the matrix

derived from the transpose of the document-to-document matrix; and the third one is an $r \times r$ diagonal matrix of singular values where r is the minimum between the row and the column of the original matrix, and the rank of the term-document association matrix. Consider that only s largest singular values of the third matrix are kept, along with their corresponding columns in the first and the third matrix, while the rest of the singular values are deleted. The resultant matrix is closest to the original matrix in the least square sense with rank s . The relationship between two documents in the reduced space of dimensionality s , can be obtained from the multiplication of the resultant matrix and its transpose. To rank documents with regards to a query, the query is modelled as a pseudo-document in the original term-document matrix. If the query is modelled as the document with number 0, then the first row in the multiplication of the resultant matrix and its transpose provides the ranks of all documents with respect to this query (Dong et al, 2008; Song and Park, 2009). (Zaman and Brown, 2010), applied LSI in order to study the three most popular weighting schemes which are TF-IDF, log-entropy and raw-term frequency to find which weighting scheme is more effective for very large data set indexed using LSI. It shows that TF-IDF is the best among these three schemes with score of 0.164 using average 11-points for precision-recall. However, when using P-R@N measure, this score increases to 0.88 for $N < 4$.

2.3.3.3 Probabilistic IR

Probabilistic retrieval is based on estimating the probability of relevance of a document to the user for a given user query. Typically, relevance feedback from a few documents is used to establish the probability of relevance for other documents in the collection (Fuhr and Buckley, 1991; Gordon, 1988). There are several probabilistic IR techniques developed in this domain. (Zhang, Wei and Meng, 2012) proposes an automated ranking approach based on probabilistic information retrieval model to solve the Many-Answers Problem of XML twig query. This approach applies the probabilistic information retrieval model to capture the correlations between the unspecified and specified values of leaf nodes as well as the user preferences based on the XML data and query history, and then it constructs the scoring function and ranks the query results according to the ranking scores. This approach evaluates the documents using a modified version of the

known TF-IDF formula and is applied on a set of XML documents including 100,000 used car elements. The achieved precision using this approach reaches to 0.79 in average. However, this approach lags in the factors used to evaluate the document as it depends on the statistical factors in addition to the probability of the relevance of the retrieved document for the query given the text features of that document.

There are three different learning strategies used in probabilistic retrieval. The first one applies estimation of probabilities of relevance to a set of sample documents (Robertson and Jones, 1976). The second one applies estimation of probabilities of relevance to a set of sample queries (Maron and Kuhns, 1960). The last one applies estimation of probabilities of relevance to all documents or queries. Inference networks (Turtle and Croft, 1990, Manning, Raghavan, Schütze, 2009) use a document and query network that capture probabilistic dependencies among the nodes in the network. The average precision achieved by last mentioned technique is 0.245.

2.3.4 Which IR Paradigms to Choose?

In spite of the extensive enhancements achieved on Web search, web users still encounter two major problems when trying to retrieve useful information (LEE, 2007; Bhatia and Khalid, 2007; Haveliwala et al, 2002). These problems are: low precision, which is due to the irrelevance of many of the search results, and low recall, which is due to inability to index all the web documents available on the Web and related to user query. Many researchers used Web information retrieval to solve these two problems, since the aim of a search engine is to retrieve all documents relevant to the user query (high recall), and not to retrieve any irrelevant document (high precision).

Considering the web, which consists of a tremendous number of documents that need to be evaluated and ranked using the information retrieval system, it is found that the Boolean model is not suitable although it is less computationally costly because it doesn't provide ranking methods where documents are marked as either relevant or not without providing a degree of relevance (Dong et al, 2008). The fuzzy set theory computes the degree of semantic relevance between two terms, which is more efficient in improving the precision. However, the cost associated with computing the relevance between two terms depends on the number of occurrences of the terms in all documents, which makes

the implementation of the fuzzy set theory in large scale databases costly (Dong et al, 2008). The extended Boolean model overcomes the limitations of the Boolean model, where the documents can be ranked by extending the relevance between the documents and the query. However, the extraction and manipulation of the index terms from dynamic sources in databases is costly on time. In addition, the cost of computing the relevance between terms is extensive (Dong et al, 2008). In VSM, the dynamic document bases make index terms difficult to maintain. In addition, the dependency of index terms is a prerequisite for VSM. Due to the locality of many term dependencies, the indiscriminate application to all documents in the collection dramatically affects the performance (Dong et al, 2008). In GVSM, the dependency is represented in an effective way to overcome the VSM dependency drawback. However, the incorporation of term dependencies does not yield effective improvement with general collections. Consequently, the GVSM does not have a clear progress in practical performance. Moreover, GVSM is more complex and computationally more expensive than VSM (Dong et al, 2008). Although LSI is an efficient indexing scheme which reduces noise and removes redundancy, it has not been validated on a large set (Dong et al, 2008; Song and Park, 2009). The best model of the probabilistic IR models is the Bayesian network model (Dong et al, 2008). It overcomes the drawback of the inference network model, as it adopts a clearly defined sample space more easily. Although it provides a separation between the document space and the query space which simplifies the modelling task and facilitates the modelling of additional evidential sources, such as past queries and past relevant information, it still has the same drawback of other techniques, which are the high cost of computation and the limitation of the factors used to measure the degree of relevance to the query.

Generally speaking, and according to Dong et al (2008), the main drawback of these traditional IR models is that they are computationally costly making them not suitable for web search domain. Moreover, it is impractical to solve web search problems using these traditional information retrieval techniques. This is because in such techniques, all documents must be processed and evaluated to produce the results related to the user query. Nevertheless, considering the huge number of web documents available in the search space, it makes this solution impractical from a time processing point of view

since the user is going to wait for very long time before he gets the results of his query. On the other hand, if the IR system is going to process a limited number of documents in order to be fast, then it could not retrieve all relevant documents and may process and retrieve irrelevant documents ending with very low recall and a very low precision percentage. These problems are among the tasks to be addressed in this work. Moreover, using neural networks to solve web search problems is impractical since this needs examples or training sets to start with before the actual process starts to generate and produce the final results (Chen, 1995), in addition, this technique fits well with conventional retrieval models such as vector space models and probabilistic model (Chen, 1995). Consequently, this technique is extensive in computation and as a result, requires a longer time to get the required results (Dong et al, 2008).

2.4 Genetic Algorithm

As mentioned in the introduction of this chapter, GA is one of the evolution based algorithms and it became an important approach when used to provide suitable information for the user's needs. Therefore, it has been adopted by many researchers to enhance recall and precision of the retrieved documents as will be seen in this section.

GA is a probabilistic algorithm used to simulate the mechanism of natural selection of living organisms. It is often used to solve problems having expensive solutions. This is basically due to the principles of selection and evolution employed to produce several solutions for a given problem. Generally speaking, GA's search space is composed of candidate solutions (chromosomes) to the problem. Each chromosome has an objective function value known as fitness value. This measure is used to favour selection of successful parents for new offspring. Offspring solutions are produced from parent solutions by the application of selection, crossover and mutation operators (Radwan et al 2006).

The most common type of genetic algorithm used for web search works as follows: a population of web pages or documents is created with a group of individuals selected randomly, normally either by randomly generating an IP address or by querying a standard search engine such as Google, Yahoo, MSN, etc, or having a predefined set of

documents such as TREC and CACM. The individuals (retrieved documents) in the population are then evaluated using what is called a fitness function. This fitness function is provided by programmers and gives the individuals a score based on how relative are they to the user query. Two individuals are then selected based on their fitness. The higher the fitness, the higher the chance of being selected. These individuals are then "reproduced" by operation called crossover to create one or more offspring, after which the offspring are mutated randomly. This continues until a suitable solution is found or a predefined number of generations are created, depending on the needs of the system. The research areas in GA tackled by researchers cover wide range of IR topics such as query induction, representation, and optimization; document clustering; and document matching and ranking. Next section is going to discuss the current work in IR in general then the focus will be on the techniques of implementing each GA operator.

2.4.1 Genetic Algorithm in IR Domain

GA is characterized by the intrinsic parallel search mechanism and powerful global exploration capability in a high-dimensional space. Therefore it is intensively used to solve a wide range of hard optimization problems that have no best known solutions. For this merit, there is an increasing interest in applying GAs to intelligent IR in recent years.

Gordon (1988) presented an approach for re-describing document descriptions and based on that he adopted a similar approach to document clustering (Gordon, 1991). (Raghavan and Agarwal, 1987) have also used GA's for modifying document clustering. (Yang et al, 1993) used GA to improve queries by using the relevance feedback. The average 11-point precision-recall achieved was 0.1213. (Aly, 2007) has also applied GA to improve queries by modifying user's queries based on relevance judgments. This method achieved average 11-point precision-recall of 0.27397. (Chen, 1995) used GA to optimize keywords that were used to suggest relevant documents. Ozel (2010) used GA to classify web pages by extracting the most important HTML tags and construct features from tag-term combination. This approach increased the document classification accuracy by 95%. (Ashraf, Ozyer, and Alhaji, 2008) use clustering of HTML documents based on set of features composed of HTML, semantic and orthographic features, combined to better represent a particular domain. This technique is applied on 18 web pages and achieves

high score of precision which is 0.9455 for precision, and 1 for recall. However, this size of data set is not enough to judge its high performance. (Dashti and Zad, 2010) used GA in a distributed way according to users' favourites to optimize query sent to the search engine to finally optimize the quality of result pages. (Saini, Sharma, and Gupta, 2011) applied implemented an IR system using GA model in which the fitness function was combination of the know Cosine measure, Jaccard measure, and the developed semantic similarity measure to form a semantic-based-combined-similarity measure. The best results achieved by this model using the precision-recall measure are ranging between 0.932 for P-R@10, and 0.19 for P-R@100. These results are obtained when applied on CISI plain document set of 1414 documents. (Yan et al, 2009) proposed a new approach of IR which is quantum-inspired GA. This approach combines GA with quantum computing principles such as quantum bit and superposition of states. This approach is investigated against 5000 document downloaded from the Web and indexed using VSM. These documents are weighted using the TF-IDF formula, and the fitness function is derived from

$$f(x) = \exp(\text{sim}(D_i, D_j))$$

In this formula, D_i and D_j are two documents which their similarities are to be compared. The precision-recall achieved by this approach ranges between 0.9 for P-R@10 and 0.07 for P-R@100. Another approach that applies GA in IR domain is the one introduced by (Xu, Deli and Yu, 2009). This approach combines GA with simulated annealing algorithm based on the vector space model. The maximum achieved score for precision-recall measure is 0.75 for P-R@10 and drops to 0.6 for P-R@100. (Sehgal et al, 2009) developed an approach to classify multiple database records such as MIDLINE and Swiss-port. This approach is implemented using genetic algorithm and compared to Handcrafted Rule-Based Classifiers. The results show that the former approach achieved higher precision which reached to 0.857.

The following subsections elaborate with more details on the techniques used to implement each one of the GA operators

2.4.2 Initial Generation Creation

As stated earlier, GA produces several generations before the optimal solution is found. However, the selection of first generation has a special importance because the characteristics of next generations are inherited from this generation. Moreover, the speed of finding the optimal solution depends on the quality of the individuals of the first generation. Therefore a particular attention is given to this stage as it strongly affects the GA process. The search space, which the initial generation is subset of, can be generated in many ways. One of which is to query any search engine, such as: Google, yahoo, MSN, etc. But for the purpose of analysis and empirical studies, most of researchers use a prepared collection set of documents such as: Text Retrieval Conference (TREC) series (Kim and Zhang, 2003; Kim and Zhang, 2000), Communication of the Association for Computing Machinery (CACM) set (Aly, 2007) and Communications of the Institute for Scientific Information (CISI) (Vrajitoru, 2000; Saini, Sharma, and Gupta, 2011). In fact, this method is commonly used by researchers because it has a prepared list of documents along with queries and their relevant documents. Selecting individuals to form first generation of genetic algorithm system from the search space can be implemented using various methods. One of these methods is to query standard search engines using heuristic creation operator to generate the initial population (Marghny and Ali, 2005). The second method is the most popular one. It involves the selection of individuals randomly from search space without any specific criteria (Pathak, Gordon and Fan, 2000; Aly, 2007; Yeh et al, 2007; Beasley, Bull, and Martin, 1993; Beasley et al, 1993a; Zhang, et al., 2005; Noreault et al, 1980; Horng and Yeh, 2000; Martín-Bautista and Vila, 1998; Lopez-Pujalte, Guerrero-Bote, and de Moya-Anegon, 2003a). Heuristic initialization is a third method used to create initial generation. It is achieved by applying some filtration on the randomly selected individuals so that search begins with some good points (Beasley et al, 1993b). Martin-Bautista and Vila (1998) select two individuals randomly and XOR them to generate new individual of initial population. (Kim, Zhang, 2000) use a document judged relevant by a user as an initial generation.

2.4.3 Assessment of Initial Generation Creation

Since this work is based on predefined set of documents, then the first option, which is concerned with starting by individuals selected from standard search engine, is omitted from consideration. When looking at other methods of creating initial generation, there is a trade-off between creating initial generation in a fast way with low quality or slower way but with high quality. Fast creation is done by selecting individuals randomly without any selection criteria. However, this method may stick at a local optimum solution causing the results to be less effective. Example of these low performance due to this technique of selection is (Aly, 2007) which achieve an average precision-recall score of 0.297. To avoid this, GA could produce larger number of generations as stated by (Yeh et al, 2007); hence we see that Aly's approach (2007) converges after 100 generations. The second method is to select individuals based on some criteria. Although this method slows down the creation of initial generation, it provides a higher probability to find optimal solutions rapidly. The former method is practical when the population size is big. Accordingly, selecting individuals randomly creates the population rapidly. However, the second method is much better when the population size is relatively small or controllable. This is because the created population starts by good points and also allows finding optimal solution faster. Hence, the proposed approach in this thesis will use the second method. The last two methods are not practical here. The first one of them (Martín-Bautista and Vila, 1998), which involves the selection of two individuals randomly and XOR them, is applicable on binary representation of individuals and this type of representation doesn't match the proposed model. The second method uses document judged relevant by a user and this involves user interaction in this process which reduces its speed of the process. Hence, it is not considered in this study. A brief description of creating initial generation along with advantages and disadvantages of each method are listed in Table 2-2.

Table 2-2: Summary of creating initial generation methods in GA systems

Creating initial generation techniques	Description	Advantage	Disadvantage
heuristic creation operator (Marghny and Ali, 2005)	outputs a web page from the results given by four standard search engines (AltaVista, Google, Msn, Yahoo)	• Fast to generate the initial generation	• Search can begin with bad points

Top 15 (Radwan et al 2006)	Top 15 documents retrieved from classical IR	<ul style="list-style-type: none"> • Search begins with some good points 	<ul style="list-style-type: none"> • Limited number of document to start with
Random selection (Pathak, Gordon and Fan, 2000; Aly, 2007; Yeh et al, 2007; Beasley et al, 1993a; Zhang, et al., 2005; Noreault et al, 1980; Horng and Yeh, 2000; Martín-Bautista and Vila, 1998; Lopez-Pujalte, Guerrero-Bote, and de Moya-Anegon, a, 2003; Drias, Khennak, and Boukhedra, 2009)	Selecting individuals randomly from search space	<ul style="list-style-type: none"> • Fast to generate the initial generation 	<ul style="list-style-type: none"> • Search can begins with bad points • Slow the process • Could finish without finding relevant documents
Selective random selection (Beasley et al, 1993b)	Applying some filtration on the randomly selected individuals	<ul style="list-style-type: none"> • Search begins with good points • Reduce process of finding optimal solution 	<ul style="list-style-type: none"> • Slow the process while filtering the documents.
Document judged relevant selection (Kim and Zhang, 2000)	collection of documents initially judged relevant by a user represents the initial population	<ul style="list-style-type: none"> • Search begins with good points 	<ul style="list-style-type: none"> • Involves user interaction. • Slow the process

2.4.4 Fitness Function

The concept of GA system is to create several generations before finding the optimal solution. These generations are obtained from initial generation by the process of selection, crossover and mutation. The individuals are selected according to their performance to participate in crossover. The performance is evaluated using fitness function. The fitness function (FF) is a performance measure or reward function, which evaluates the relevance of the document to the user query. During the GA process, the fitness function is used in two operators. These operators are selection and mutation. Figure 2-2 shows where the fitness function is applied to the GA process where it is presented in bold diamonds.

Fitness functions used in GA -as it is noted - are of three types: The first type either consists of summation of term weights or it has the term weight as a main component. The second type of fitness functions is the similarity measure. The third type is a customized fitness function in which authors develop their own fitness functions that suit their GA system. To understand the first type of fitness function there is a need to explain

the term weight.

2.4.4.1 *Term Weight*

It is a score assigned to a term reflecting its importance within the document. It forms one of the main factors in fitness function used by researchers as noted in the literature. The weight of the term within a document or among a collection depends on many factors. These factors are either local factors (within document) or global factors (collection wide). Famous local factors used are term frequency within a document (Kim and Zhang,

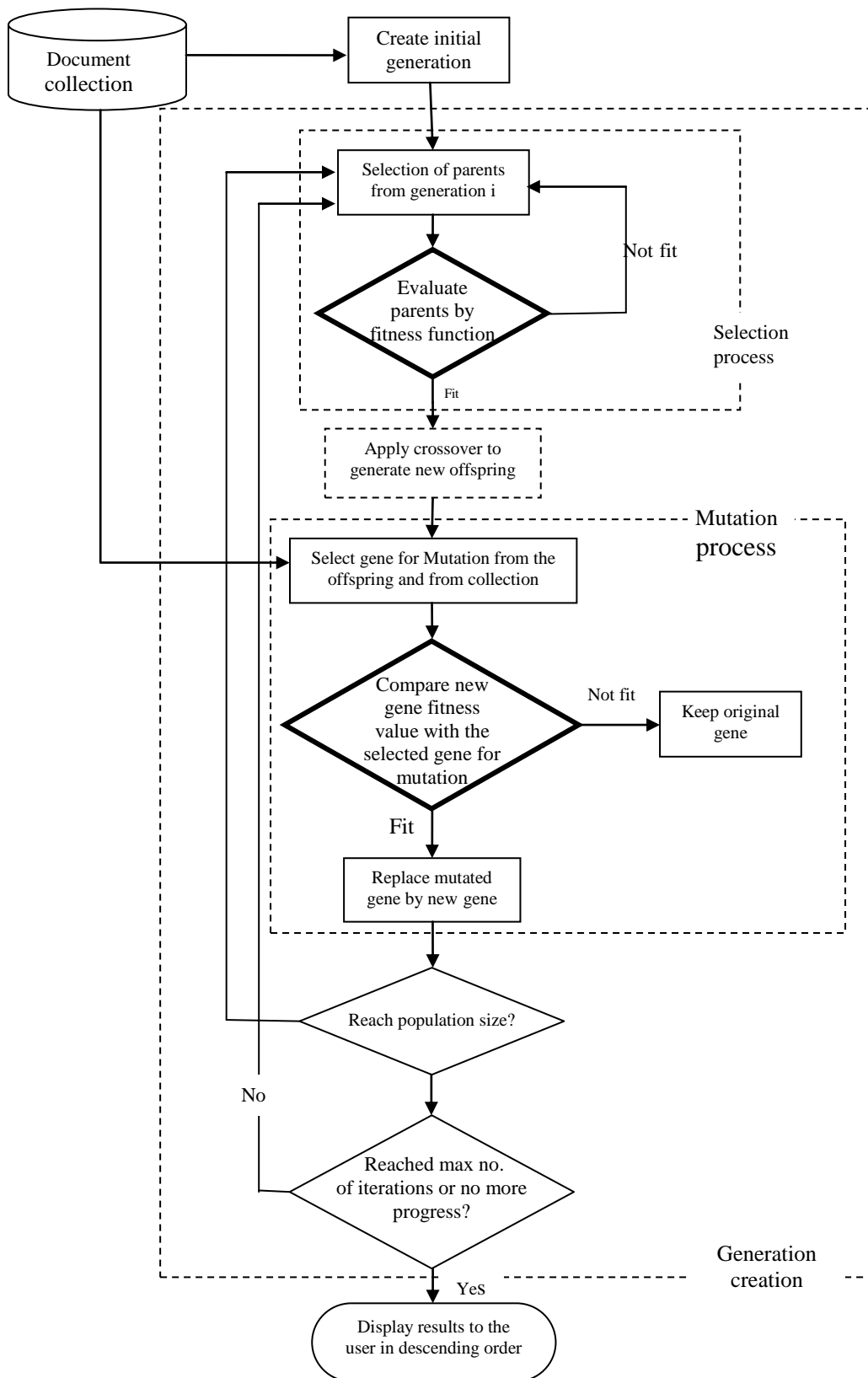


Figure 2-2: GA process showing the role of fitness function highlighted by bold diamonds

2003; Radwan et al 2006; Cummins and O’Riordan, 2006; Salton and Buckley, 1988), document size (total terms frequency) (Kim and Zhang, 2003; Cummins and O’Riordan, 2006), frequency of the most frequent term within the document (Kim and Zhang, 2003; Radwan et al 2006; Aly, 2007; Vrajitoru, 2000), and number of unique terms within the document (Cummins and O’Riordan, 2006). Global factors used in evaluating term weight are: number of documents referencing a term (Kim and Zhang, 2003; Radwan et al 2006; Billhardt et al, 2002; Vrajitoru, 2000; Cummins and O’Riordan, 2006; Salton and Buckley, 1988), total number of documents in the collection (Kim and Zhang, 2003; Radwan et al 2006; Billhardt et al, 2002; Vrajitoru, 2000; Cummins and O’Riordan, 2006), frequency of the term in the collection (Cummins and O’Riordan, 2006), and total number of terms in the collection (Cummins and O’Riordan, 2006).

The simplest formula of term weight is the one that considers the term frequency within the document as a term weight. But most common term weight formula used by researchers (Cummins and O’Riordan, 2006; Xu, Deli and Yu, 2009) is the one proposed by Salton and Buckley (Salton and Buckley, 1988). It is multiplication of Term Frequency by Inverse Document Frequency (TF-IDF). In this approach the weight of term i in document j is defined as:

$$w_{ji} = tf_{ji} \times idf_{ji} = tf_{ji} \times \log_2\left(\frac{N}{df_{ji}}\right)$$

where tf_{ji} is the number of occurrences of term i in the document j ; df_{ji} is the frequency of documents referenced by term i in data set and N is the total number of documents in the collection. Although it uses limit factors to evaluate the term, still many users apply it in their researches because of its popularity and been accepted by the researcher. However, using this formula alone may reduce the quality of retrieved documents.

Vrajitoru (2000) uses normalized term frequency and normalized inverted frequency although the author didn’t justify the advantage of this method over classical TF-IDF method. Normalized term frequency is computed as the actual frequency of a term within the document divided by the frequency of the most frequent term in that document. Normalized inverted frequency is defined as $(\log(N) - \log(df_i)) / \log(N)$, where N is the total number of documents in the space. This approach achieved best results for average

precision-recall of 0.383 which is relatively low.

Probabilistic version of TF-IDF is called Okapi-BM25 and was developed by Robertson et al. (1998). It was shown that it achieves a higher average precision than TF-IDF on large document collections. (Cummins and O’Riordan, 2006) compare the three weighting schemes, namely, the classical TF-IDF, the Okapi-BM25 and their own developed formula which is:

$$w_t = \frac{\frac{cf}{df} \times (\log(tf) + \frac{cf}{df})}{2df + l + tf}$$

Where cf is the frequency of a term in the collection, df is the number of documents containing the term, tf is the term frequency in the document and l is the document length. The results show that TF-IDF has the lowest performance where it achieved an average score of 35.86%, while the above formula achieved 38.85, and OKAPI-BM25 achieved 39.4%.

Cummins and O’Riordan (2006) found that adding more terminals to the weighting function increases the precision until reaches maximum when all the above described terminals are forming the upper mentioned formula. Also they concluded that cf measure plays an important role in determining the relevance of documents in the collection they used. In addition, they combine three separate weighting schemes, namely, local weigh, global weight and query weight to form a general weighting scheme. Using this approach, they show that complete weighting scheme can outperform the BM25 weighting scheme on collections similar in size to the training set . However, the full weightings evolved on small collections do not outperform BM25 on large collections. Both Bayesian inference network model and 2-Poisson model use the same factors used in the previously mentioned term weight except that 2-Poisson model uses additional factor, which is the average document length in the collection (Kim and Zhang, 2003).

Considering HTML documents, (Marques Periera, Molinari, and Pasi, 2005) proposed a weighting scheme for non-linear contextual model. This approach considers the total number of meaningful HTML tags and assigns weight for each tag such that weight of 1

is assigned to the most important tag and n to the least important tag. Subsequently, it considers the total length of each tag by summing the number of terms within the tag. Then, from these 2 factors, a weighting formula is constructed in a way that it favours terms that appear in higher important tag and occur in a shorter length tag. The advantage of this approach is the adaptive determination of the weight assigned to the keywords. This adaptation allows considering “to some extent” the author’s writing style that is hidden in the HTML tag distribution. On the other hand, this approach requires extensive computational time to calculate the term weight since it requires parsing the whole document to obtain the total number of tags. To find the length of each tag requires parsing it again to assign the weight for each term accordingly. In another word, the time complexity is $O(n^2)$. Moreover, it treats all documents having the same number of tags equally regardless of the tag importance. For example: if a document has *title*, *Header1* and *anchor* tags and another document has *header3*, *bold* and *italic* tags, then this approach will assign same term weight for a word appearing in the first tag, while logically, the former document must have higher weight since *title* tag reflects the content more clearly than *header3* does.

2.4.4.2 Similarity Function

The second type of fitness function is the similarity function. It measures how close is the document vector to the user query vector. In other words, it measures the distance between the document and the query vector. Some authors use similarity measure as a fitness function and others use it as a component in the fitness function. Most common similarity function used as a fitness function is the cosine similarity measure (Radwan et al 2006; Aly, 2007; Vrajitoru, 2000). It is found that the average precision-recall score achieved by (Radwan et al 2006) is 0.437, 0.432 by (Aly, 2007) and 0.383 by Vrajitoru, 2000). On the other hand, Wang and Feng (2005) use cosine similarity as a component of the Fitness function. It is represented by the ratio of summation of similarity measure of set of relevant documents retrieved to the summation of similarity measure of set of non-relevant documents retrieved; all are divided by the total number of documents. This approach is evaluated by comparing the number of relevant documents after each iteration, where it ranges from 100% at the first one and it reaches to 10.8% at the fifth

iteration. Normally, the performance of GA increases as more iterations are performed, while this approach has opposite behaviour.

Other similarity measures are Dice coefficient (Klabbankoh and Pinngern, 2008), Cosine coefficient (Klabbankoh and Pinngern, 2008) and Jaccard coefficient (Klabbankoh and Pinngern, 2008) (these formulas are listed in Table 7-1 in appendix A). However, these measures are most suitable for systems implemented using vector space model, because in this model both the document and the query are represented as vectors making applying similarity measures simple and straightforward. In this study, comparison is done between these three formulas when applied by GA on document set of 343 documents. The results are analyzed using F1 measure where:

$$F1 = \frac{2 \times Recall \times Precision}{Recall + Precision}$$

The achieved results are 0.8625, 0.89625 0.89125 for Dice coefficient, Cosine coefficient and Jaccard coefficient respectively. These results reflect the high performance of these formulas which prove that these are better to be considered in case of vector space model of implementation. Another point worth mentioning is that the document is very small to compare the performance with other formulas.

2.4.4.3 Custom Fitness Functions

This type of fitness functions is the third type developed by researchers to evaluate a chromosome. One of these fitness functions that is applicable to the vector space model is proposed by Radwan et al (2006). It evaluates the difference between terms weight of a given chromosome and the query vector. The complexity of this fitness function is n^2 as compared to the complexity of n^5 of the cosine similarity fitness function, where n is the number of terms in the search space for the query, and has a precision value better than that of the cosine similarity fitness function as stated by the author (formula 6 in table 7-2). This approach achieved a maximum average precision-recall of 0.437 which applied on CISI document se which consists of only 1414 documents.

Another formula is the one developed by Marghny and Ali (2005) which evaluates the document based on link quality within the document and uses the summation of mean

number of query keywords occurrences in links. This function considers the terms that appear in the links within the document only regardless of other factors that may affect document evaluation; hence, the performance of the document is evaluated based on limited factors. This approach is analyzed in terms of the mean quality of the population and the results show that the average precision reaches 0.2531 in average after 1000 iterations.

Another technique called Geniminer proposes a method of GA based on a fitness function developed by Picarougne et al (2002a). It results in a high score if the document contains many keywords given by the user with uniform proportion of each keyword, many words from the list of words that must exist and no words from the list of words that must not exist and many links that might lead to relevant pages (formula 7 in table 7-2). This approach is tested by 10 queries applied on 500 documents. The evaluation is done by analyzing the average population mean quality for different mutation rate, and it is found that the best is achieved when mutation rate is 0.5. At this point, the population size is 100 and the mean quality is 457.9.

Honng and Yeh (2000) used non-interpolated average precision as a fitness function. It is similar to the average precision but with cut-off points equivalent to the training documents. (Kim and Zhang, 2003) used two weighting models in addition to the classical Salton model. The first of these methods is the Bayesian inference network in which the factors are term frequency per document and frequency of the most frequent term in the document. The second weighting model is the 2-Poisson model. The factors used in this model are term frequency per document, document length, average document length, total number of documents in the collection, and the number of documents in which a term occurs. With these fitness functions the achieved precision at 10 top retrieved documents for Bayesian is 0.45 and for 2-Poisson model is 0.63.

The fitness function used in (Húsek et al 2005) is called precision fitness and is composed of two parts added together: the first part reflects recall quality and the second part reflects precision quality. Influence of each part is given by α and β coefficients in the precision fitness function. The precision achieved by this approach is 0.75 while the recall is 1. In fact these are very attractive results. However, this system was run on 5000

documents for 1200 iterations before achieve this. In fact, this huge number of iterations reflects somehow the weakness of this technique compared with others.

In (Marghny and Ali, 2005), the fitness function for a web page (web document) is composed of link quality and page quality. The link quality is expressed as:

$$F(L) = \sum_{i=1}^n \#K_i,$$

Where: n is the total number of input keywords, $\#K_i$ is the number of occurrences in link, and $K_1, K_2, K_3 \dots$ are the keywords given by the user.

The page quality is determined as:

$$F(P) = \sum_{j=1}^m F_j(L),$$

Where m is the total number of links per page.

Then the fitness function is the mean quality and is expressed as:

$$M_q = \frac{F_{max}(P) + F_{min}(P)}{2}.$$

This model was applied to a sample of 250 pages using 10 queries to evaluate the GA effect on the quality of the results. This evaluation is done using the population averaged mean quality for different values of population size that forms 25%, 50%, 75% and 100% of the pages retrieved from standard search engine. Their findings show that small size of pages and small size of population limit the chances of improving the page qualities and reduces the execution time at a specified number of iterations. This technique was evaluated using the average mean quality. It shows that the best average precision of 25.31% is achieved at 50% of population size, and this was achieved after 10000 iterations which is impractical long time for document retrieval compared to other studies who present the results with much fewer generations such as 50 generations by (Cummins and O’Riordan, 2006).

For HTML documents, (Kim and Zhang, 2003), evaluate the similarity of a document to

the query by assigning a weight for each HTML tag, then compute the similarity of document d to the query q using the following formula:

$$Sim(d, q) = \sum_{k=1}^T \alpha_{dk} \cdot w_{dk} \cdot w_{qk}$$

Where α_{dk} is the weight of term k , w_{dk} is the weight of the k^{th} term in the document d , w_{qk} is the weight of k^{th} term in the query q , and T is the number of terms. This approach applies GA to determine the proper weight that can be assigned to each HTML tag. And then the results are analyzed by comparing the precision of the output with and without applying the HTML tag. The best score was 0.35 for P@10. However, these results are produced using the classical TF-IDF to weight the terms.

(Marques Periera, Molinari, and Molinari, 2005) utilize the frequency of terms within the HTML tag to evaluate the degree of significance for a term t . Given a term t_i in document d belonging to a class of tags $ctag$, then the significance degree $F(d, t)$ is computed as:

$$Sim(d, t) = \left(\sum_{i=1}^n w_i t_i \right) g(IDF_t)$$

Where w_i is the numerical importance weight $w_i > 0$ that is assigned to each $ctag$, such that $\sum_{i=1}^n w_i = 1$, and $g(IDF_t)$ is the normalized inverse document frequency. This technique was applied on two documents only having 9 classes of HTML to analyze the term significances. Hence the results are inadequate to be criticised.

The last fitness function to be mentioned in this section is the one developed by (Saini, Sharma, and Gupta, 2011). This function combined three measures, named: Cosine measure, Jaccard measure and the semantic similarity measure, where in the last measure, a weight assigned to each term which can be a single word or phrase is based on the count of how many times a term is used as an argument in the whole document in every verb argument structure of sentences. This function is represented as:

$$SBCSM = W1 * \text{similarity-measure} + W2 * \text{cosine-similarity} + W3 * \text{Jaccard-measure}$$

This measure is applied on CISI document set which composed of 1414 plain text documents. The results obtained using precision-recall measure ranges between 0.932 P-R@10 and 0.19 for P-R@100.

2.4.4.4 Assessment of Term Weight and Fitness Function

Investigation of the above mentioned fitness functions reveals that they use common factors. These factors are term frequency in the document, global term frequency in document collection, total number of documents in the collection, and number of documents that reference certain term. Some other factors which are used by some researchers in this area are the frequency of the most common term in a document, global frequency of most common term in document collection, document length, average document length within the collection, total number of terms in the collection, total number of unique terms in a document, and in the document collection. These factors are more statistically than being content descriptive. Hence they reflect one side of the relativity of the documents. Hence, it is noted that the average precision-recall achieved ranges between 0.297 (Aly, 2007) and 0.75 (Xu, Deli and Yu, 2009). Moreover, several approached analyzed are applied on limited document set of size 343 (Klabbankoh and Pinngern, 2008), 500 by (Picarougne et al, 2002), 1033 by (Cummins and O’Riordan, 2006) and 1460 by (Vrajitoru, 2000; Radwan et al, 2006; Aly, 2007).

Although (Húsek et al 2005) achieved high recall (100%) and high precision (75%), but this is after runing the system for 1200 iterations making time performance far from other approaches, where other approaches generate lower number of iteration such as (Radwan, 2006; Aly, 2007) generated 100 generations, (Vrajitoru, 2000) generated 40 generations, and (Cummins and O’Riordan, 2006) generation 50 generations.

Moreover, some of these factors are dependent on the collection set when considering the global factors such as the total number of terms in the collection, the total number of documents in the collection, the most frequent term and the number of unique terms in the collection. That means, when considering one document as relevant, implies that it is relevant compared to other documents in the collection, but could not be purely relevant. In addition, they are more suitable for plain text rather than structured or semi-structured documents. Additional or sometimes alternative factors are required to be considered

when evaluating the document relevance to the user query.

Considering HTML documents which form most of web documents (Kim and Zhang, 2003), HTML tags play a vital role in determining term importance. If the term appears in the title or some headers within the document with low frequency then it may reflect the content of the document more accurately than terms that appear more frequent but somewhere close to the end of document or in the body of the document.

Another factor that needs to be included in evaluating the document is the number of the query keywords that appear in the document. If a document includes all query keywords then it is more relevant than document that has subset of the query keywords. The third factor that needs to be included in evaluating the document is the distance between query terms. Finally and not last is the first appearance of the query keywords within the document. It is assumed that more relevant documents refer to user query terms in the first few sentences. Although this factor is not always true, including it in the fitness function is expected to enhance the evaluation process. List of term weight formulas and fitness functions formulas used by authors in GA systems are included in appendices A and B.

2.4.5 Parents Selection

The main three operators of GA that produce next generations are parent selection, crossover and mutation. Parent selection is controlled by the fitness function which favours certain individuals based on their fitness value. Parent selection forms a central component of the genetic algorithm (GA) (Holland, 1975; Goldberg, 1989) and directly controls the exploitation factor in the “exploitation-versus-exploration” trade-off that is believed to be critically important in the working of the GA (Chakraborty, Deb, and Chakraborty, 1996). As a preparation for crossover, parents need to be selected such that the selection operator is intended to improve the average quality of the population. This can be achieved by giving individuals of higher quality a higher probability to be copied into the next generation based on the assumption that better individuals are more likely to produce better offspring (Kim and Zhang, 2003).

Several selection methods were developed. The simplest one is to select parents

randomly without any restriction or evaluation (Martín-Bautista and Vila, 1998). It is obvious that this method causes poor selection since it doesn't favour the fittest individuals and hence it is rarely used. However, there are several commonly used methods for selection. The most popular one is *simple random sampling selection* also called *proportional selection*. It has been applied by many researches (Petridis, Kazarlis and Bakirtzis, 1998; Chen, 1995; Pathak, Gordon and Fan, 2000; Billhardt et al, 2002; Vrajitoru, 2000; Kim and Zhang, 2003; Lopez-Pujalte, Guerrero-Bote, and de Moya-Anegon, 2003a; Lopez-Pujalte, Guerrero-Bote, and de Moya-Anegon, b, 2003; ; Radwan et al 2006; Saini, Sharma, and Gupta, 2011; Pandey, Dixit and Mehrotra, 2012) and was also recommended by Goldberg (1989). This method performs roulette-wheel selection, where each individual is represented by a space that proportionally corresponds to its fitness. By repeatedly spinning the roulette wheel, individuals are chosen by using stochastic sampling.

Another common method for selection is the *tournament selection* used by Holland (1975) and by Yeh, Lin, Ke and Yang (2007). In this approach a group of i individuals are randomly chosen from the population. This group takes part in a tournament and an individual with highest fitness value wins. In many cases i is chosen to be two, and this method is called *binary tournament selection*. In ranking selection the individuals are sorted according to their fitness values and rank N is assigned to the best individual and a rank 1 to the worst (Goldberg and Deb, 1991; Húsek et al, 2005). The selection probability is linearly assigned to the individuals according to their rank. In truncation selection (Yang, Korfhage, and Rasmussen, 1992; Muhlenbein and Schlierkamp-Voosen, 1993) with threshold t and the fraction f of best individuals are selected and mated randomly with the same probability until the number of offspring is equal to the size of the population. Genitor selection (Goldberg and Deb, 1991) works individual by individual, choosing an offspring for birth according to linear ranking, and choosing the currently worst individual for replacement.

Tate and Smith (1995) Suggest another selection method by selecting a uniform random number between 1 and \sqrt{S} , where S denotes the population size, and then squaring it. The result is truncated and taken to be the rank of the selected parent. If the result is one, the

best fitting individual in the population will be chosen. However, Tate and Smith (1995) didn't provide justification or analysis for using this method. On the other hand, a comparison between several parent selection methods in terms of time complexity was performed by Goldberg and Deb (1991). It shows that tournament selection, Stochastic remainder proportionate, and Stochastic universal proportionate have the lowest time complexity of $O(n)$, where roulette-wheel, ranking and Genitor have complexity of $O(n \log n)$. On the other hand, Genitor selection and overlapping population selection show higher growth ratio than other methods.

Another advantage of *simple random sampling* over *proportional selection* method rather than time complexity is that there is a chance for some weaker solutions to survive in the selection process. These weak solutions may include some components which could prove usefulness following the recombination process (Goldberg, 1989). In addition to that, the truncation selection may stick at local optima and cannot converge from initial selected chromosomes. Elitism is used by (Asllani and Lari, 2007; Billhardt et al, 2002) where best l members from the old generation are assigned to the new generation to ensure gradual improvement of the solution. Then the remaining members of the new population are selected using one of the above selection techniques.

2.4.6 Assessment of Parent Selection Technique

As a conclusion from the above judgment, there is a need to have a selection technique that combines the advantages of each method such as low time complexity, the capability of selecting healthy parents for crossover operation while passing some good individuals to the next generation (elitism), provided that the selected parents do not lead to local optima and do not converge at low performance. The methods of parent selection are summarized with their advantages and disadvantages in Table 2-3.

Table 2-3: Parent Selection techniques used in GA and their advantages and disadvantages

Parent selection method	Description	Advantage	Disadvantage
Pure random selection (Martín-Bautista and Vila, 1998)	Select parents randomly without any restriction	• Fast to create chromosome	• causes poor selection
Simple random sampling selection or roulette-wheel (proportional selection)	Each element has probability of selection proportional to its fitness	• High fitted individual are selected with high	• Certain parents may be selected frequently causing fast

(Kim and Zhang, 2003; Radwan et al 2006; Pathak, Gordon and Fan, 2000; Billhardt et al, 2002; Vrajitoru, 2000; Lopez-Pujalte, Guerrero-Bote, and de Moya-Anegon, a, 2003; Lopez-Pujalte, Guerrero-Bote, and de Moya-Anegon, b, 2003; Petridis, Kazarlis and Bakirtzis, 1998; Chen, 1995; Goldberg, 1989; Saini, Sharma, and Gupta, 2011, Pandey, Dixit and Mehrotra, 2012)		probability.	convergence. <ul style="list-style-type: none"> • High time complexity • Requires evaluation of all population to specify the probability of selection.
Tournament selection (Yeh, Lin, Ke and Yang, 2007; Holland, 1975)	Group of individuals are randomly chosen and the one with highest fitness value wins	<ul style="list-style-type: none"> • Wide range of individuals are selected • Fit parents are selected • Low time complexity • Allow week individuals to participate in the solution • High growth ration with large tournament size 	<ul style="list-style-type: none"> • Slower than random selection
Ranking selection (Goldberg and Deb, 1991; Húsek, Snašel, Owais, and Krömer, 2005)	It works by Sorting the population from best to worst, assign the number of copies that each individual should receive according to a non-increasing assignment function, and then perform proportionate selection according to that assignment.	<ul style="list-style-type: none"> • High N Fit parents are selected causing fast divergence. 	<ul style="list-style-type: none"> • Sorting reduces speed performance • Lower growth ration
Truncation selection (Yang, Korfhage, and Rasmussen, 1992; Muhlenbein and Schlierkamp-Voosen, 1993; Xu, Deli, and Yu, 2009)	The L % best individuals are selected from previous generation for mating. Normally L is in the range of 50% to 10%.	<ul style="list-style-type: none"> • Reduce time of creating offspring 	<ul style="list-style-type: none"> • Low fit parents are excluded which may contain some good individuals. • Certain parents may be selected frequently causing fast convergence.
Genitor selection (Goldberg and Deb, 1991)	It examines individual by individual, chooses an offspring for birth according to linear ranking, and choosing the worst individual for replacement	<ul style="list-style-type: none"> • High growth ration 	<ul style="list-style-type: none"> • High time complexity
Stochastic remainder proportionate (roulette)	Divide fitness of individual by the average fitness of the	<ul style="list-style-type: none"> • Low time complexity 	<ul style="list-style-type: none"> • Requires additional processing to specify

wheel selection) (Goldberg, 1989; Goldberg and Deb, 1991)	population, then the integer part of the result represents the number of times this individual is assigned as parent and the free places are filled based on roulette wheel selection (Sivaraj and Ravichandran, 2011)	(Goldberg and Deb, 1991) <ul style="list-style-type: none"> • greatest probability of selection is given to the most fit members of the population 	the probability of selecting each individual. <ul style="list-style-type: none"> • Low performance individuals have low chance to be selected
Stochastic universal proportionate (Goldberg and Deb, 1991; Yan et al, 2009; Simon and Sathya, 2009)	It is performed by sizing the slots of a weighted roulette wheel, placing equally spaced markers along the outside of the wheel, and spinning the wheel once; the number of copies an individual receives is then calculated by counting the number of markers that fall in its slot.	<ul style="list-style-type: none"> • Low time complexity 	<ul style="list-style-type: none"> • Requires knowledge of fitness value of all population prior to selection which adds extra processing load
Elitism (Asllani and Lari, 2007; Billhardt et al, 2002)	best n members from the old generation are assigned to the new generation	<ul style="list-style-type: none"> • Ensure gradual improvement of offspring • Reduce time of finding optimal solution 	<ul style="list-style-type: none"> • Used to create portion of the offspring only when applied alone.

2.4.7 Crossover Operator

Once the individuals are selected using the selection operator, they are ready for crossover operation. In GA, crossover is the second operator which is applied with a pre-defined probability to two selected individuals of a population to generate new offspring of new generation. These offspring inherit some features from parents. Higher fitness chromosome has an opportunity to be selected more than lower ones, so good solution always lives to the next generation (Radwan et al 2006; Aly, 2007).

Many algorithms apply crossover operator with a probability ranging from 0.6 to 0.8 (Minaei-Bidgoli and Punch, 2003; Picarougne et al, 2002b; Radwan et al 2006; Aly, 2007; Cutler et al, 1999; Martín-Bautista and Vila, 1998), and chromosomes not subjected to crossover are passed to the next generation and remain unmodified, where in some other systems like the one developed by (Húsek et al, 2005) it is always performed to generate the offspring. From the literature, there is a wide range of crossover techniques. However, two of them are common. These are the one-point crossover and the two-point crossover, while others are not. The first two are explained in separate subsections and the remaining is in the preceding sub-section.

2.4.7.1 One-point crossover

The simplest and most popular crossover technique is the one-point crossover (Marghny and Ali, 2005; A. Asllani and A. Lari, 2007; Radwan et al 2006; Aly, 2007; Yeh et al, 2007; Húsek et al, 2005; Billhardt et al, 2002; Vrajitoru, 2000; Vrajitoru, 1998; Beasley et al, 1993a; Desjardins, Godin, and Proulx, 2005; Martín-Bautista and Vila, 1998; Klabbankoh and Pinngern, 2008; Lopez-Pujalte, Guerrero-Bote, and de Moya-Anegon, 2003a; Song and Park, 2009; Chen, 1995). It works by choosing single point randomly within the chromosome and copy the values of parents 1 and 2 before or after this point to the same locations in the new offspring 1 and 2. Then, the values after or before this point are exchanged by copying them to the new offspring such that genes of parent 1 are copied to offspring 2 and that of 2 are copied to offspring 1. The drawback of this method is that best building blocks can be broken. Also the offspring may have lower performance than parents unless there is restriction on exchanging the genes. The third drawback is that if the cross point happen to be close to one edge of the chromosome then the generated offspring will be very similar to the parent which may delay finding the optimum solution or may fall into local optima. One way to overcome the last drawback is to apply restricted crossover where the cross point is chosen to be between the first and the last positions where the parents' chromosomes are different (Vrajitoru, 1998). Example of this technique of crossover is illustrated in Figure 2-3. In this example a crosspoint is selected randomly to be at position 6. The genes to the right of this position are exchanged between chromosome i and j and the genes to the left of this point are copied to the corresponding offspring.

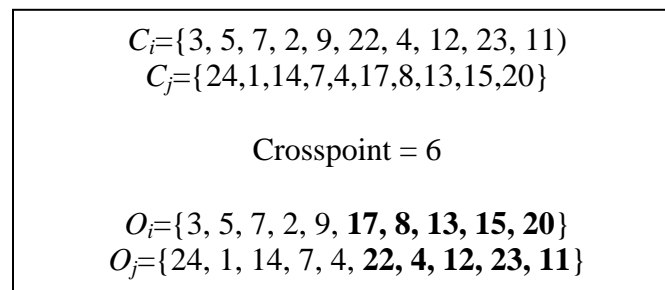


Figure 2-3: Example of one-point crossover

The results obtained by the aforementioned techniques can be summarized as follows: (Song and Park, 2006) achieved a precision of 0.755 and recall of 0.731 using a set of

500 documents.

2.4.7.2 Two-point and N-point Crossover

Another technique used to overcome the last drawback of 1-point crossover is known as the two-point crossover (Pathak, Gordon and Fan, 2000; Beasley et al 1993b; Yang, Korfhage, and Rasmussen, 1992; Spears and De Jong, 1991; Atsumi, 1997). It is similar to 1-point crossover except that 2 points are selected randomly as crosspoints and genes between them are exchanged to form the offspring. This technique provides wider diversion from parents than 1-point crossover, and researchers agree that 2-point crossover is generally better than 1-point crossover (Beasley et al, 1993b). However, if the crosspoints are close to each other then the offspring will not much differ from the parents. This technique is generalized by introducing *n*-point crossover (Vrajitoru, 1998; Klabbankoh and Pinngern, 2008; Kazarlis et al, 2001; Spears and De Jong, 1991). In *n*-point crossover the operation is done by randomly choosing a number of cross points and applies *n* simple crossover operations on the parents simultaneously. However, adding more crosspoints affects the speed of the crossover process and also disrupts the building blocks and doesn't guarantee that the offspring are better than parents although it provides wider diversity of genes in offspring. Example of 2-point crossover is shown in Figure 2-4. In this example two crosspoints are selected randomly at position 3 and 7. The genes between these two positions are exchanged between the parent *i* and *j* and the genes outside this range are copied to the corresponding offspring, i.e. genes of C_i are copied to offspring O_i and that of C_j are copied to offspring O_j .

$$\begin{array}{l}
 C_i = \{3, 5, 7, 2, 9, 22, 4, 12, 23, 11\} \\
 C_j = \{24, 1, 14, 7, 4, 17, 8, 13, 15, 20\} \\
 \\
 \text{cp1} = 3, \text{cp2} = 7 \\
 \\
 O_i = \{3, 5, \mathbf{14, 7, 4, 17, 8}, 12, 23, 11\} \\
 O_j = \{24, 1, \mathbf{7, 2, 9, 22, 4}, 13, 15, 20\}
 \end{array}$$

Figure 2-4: Example of two-point crossover

2.4.7.3 Other Crossover Techniques

Very similar to n-point crossover is the *uniform crossover* which is applied by (Cutler et al, 1999; Beasley et al, 1993b; Yang, Korfhage, and Rasmussen, 1992). It is implemented in two ways. The first one is to generate binary mask randomly with the same number of components of the chromosome. Each mask is used to generate a child from a pair of parents. The binary values zero or one in each mask are used to select the value of genes from either the first or the second parent, respectively. Example of this method is illustrated in Figure 2-5 where genes at positions corresponding to 1 in the mask are exchanged and others are left unchanged.

$$\begin{array}{l}
 C_i = \{2, 3, 4, 5, 7, 9, 11, 12, 22, 23\} \\
 C_j = \{1, 4, 7, 8, 13, 14, 15, 17, 20, 24\} \\
 \\
 \text{Mask} = \{1, 1, 0, 1, 1, 0, 0, 1, 0, 0\} \\
 \\
 O_i = \{1, 4, 4, 8, 13, 9, 11, 17, 22, 23\} \\
 O_j = \{2, 3, 7, 5, 7, 14, 15, 12, 20, 24\}
 \end{array}$$

Figure 2-5: Example of uniform crossover

The second way of implementing *uniform crossover* is to define a swapping probability p_{swap} and perform swapping between parents if the generated random is less than p_{swap} for each genes. In this technique the offspring contains a mixture of genes from each parent. The advantage of this technique is that offspring have high chance to differ from parents which makes finding the optimal solution faster. However, if the building blocks are required then this technique is harmful since the blocks are destroyed in this kind of crossover. Rather than just exchanging genes between parents to form offspring, arithmetic operations can be done on genes to produce offspring. Example of such arithmetic operations is the crossover technique called *recint* which is applied by (Minaei-Bidgoli and Punch, 2003; Yan et al, 2009). This method is applicable to chromosomes having their genes represented in real-value number but not applicable to our model where the genes are represented in integer referring to the document index. It performs an intermediate recombination between pairs of individuals where it combines

parent values to produce offspring using the formula:

$$\text{Offspring} = \text{parent1} + \text{Alpha} \times (\text{parent2} - \text{parent1}),$$

Where *Alpha* is a Scaling factor normally ranging between [-0.15, 1.55]. Bear in mind that offspring here have a visible chance to be similar to parents. However, the author used this method in his research without justification of its benefits or advantages. Another technique used to generate offspring using arithmetical operation is called *arithmetical crossover* (Kim and Zhang, 2003). Offspring in *arithmetical crossover* are generated by assigning the average of two parents for each location to the corresponding location of the offspring. One of the drawbacks of this technique is that it could produce offspring having duplicate genes where the same average may be produced from different combinations. This will lead to weak solution by diverting from the optimal solution. Another drawback of this technique and the former one is that the generated offspring could be out of the search space where the outcome of the formula could produce invalid values.

Inversion is another technique used for crossover (Beasley et al, 1993b; Goldberg, 1989) in which the order of genes between 2 randomly chosen positions within the chromosome are reversed. Hence it is applied to a single parent to produce a single offspring. However, this technique is useful when the order of genes is important; otherwise, the chromosome will remain having the same list of genes and this technique will be useless (Beasley et al, 1993b). Example of this technique is depicted in Figure 2-6. In this example, the chromosome C_i has ordered genes, then two crosspoints are selected randomly to be 4 and 8. The order of genes between these two points is reversed.

Similar to inversion is the *reordering* crossover technique but it is applied to two parents to produce two offspring (Beasley, Bull, and Martin, 1993b; Yang, Korfhage, and

$C_i = \{ 2, 3, 4, 5, 7, 9, 11, 12, 22, 23 \}$ $\text{cp1}=4, \text{cp2}=8$ $O_i = \{ 2, 3, 4, \mathbf{12, 11, 9, 7, 5}, 22, 23 \}$

Figure 2-6: Example of inversion crossover.

Rasmussen, 1992). It is applied to 1-point or 2-points crossover where the order is maintained after each crossover. The purpose of *reordering* is to find gene ordering which have better evolutionary potential (Beasley, Bull, and Martin, 1993b). It is useful when needing to discover good gene ordering but requires longer time to sort genes each time the offspring is created, and consequently causes delay to the GA process. Example of combining reordering crossover and 2-point crossover is shown in Figure 2-7. In this example, the genes between positions 4 and 8 are exchanged between parents C_i and C_j , while genes outside this window are copied unchanged. After that the genes are ordered in ascending order based on the fitness value.

$$\begin{array}{c}
 C_i = \{2, 3, 4, 5, 7, 9, 11, 12, 22, 23\} \\
 C_j = \{1, 4, 7, 8, 13, 14, 15, 17, 20, 24\} \\
 \\
 cp1 = 4, cp2 = 8 \\
 \\
 O_i = \{2, 3, 4, 5, \mathbf{13, 14, 15, 17}, 22, 23\} \\
 O_j = \{1, 4, 7, 8, \mathbf{7, 9, 11, 12}, 20, 24\} \\
 \\
 \text{After reordering: } O_j = \{2, 3, 4, 5, 13, 14, 15, 17, 22, 23\} \\
 O_j = \{1, 4, 7, 7, 8, 9, 11, 12, 20, 24\}
 \end{array}$$

Figure 2-7: Example of combining reordering crossover and 2-point crossover

It is not necessary that crossover produces two offspring from two parents. On *fusion crossover* (Vrajitoru, 1998) only one offspring is generated from two parents where for each gene, the child inherits the value from one or the other of the parents with a probability according to its performance. The advantage of this technique is that the good genes of both parents are inherited simultaneously to the offspring producing high quality offspring- in case where the probability is high enough -which speeds up finding the optimal solution. On the other hand, the generation of each population will take double of the time elapsed in normal techniques in order to generate same size of population.

Another crossover technique is used in (Vrajitoru, 1999) called *Dissociated* crossover. It uses 2-point crossover, but applies a different 1-point crossover operator to each parent. The First offspring is generated as follows: Genes in position less than the first crosspoint are copied as is from parent 1, and genes between two crosspoints are copied either from

parent 1 or parent 2 (the author didn't mentioned a criteria to select from which parent). Genes after the second crosspoint are copied from the second parent. The innovation of this technique is the creation of the second offspring where genes before the first crosspoint are copied from parent 2 and genes after the second crosspoint are copied from the first parent and genes between crosspoints are replaced with 0 in binary representation. The author compared this technique with classical 1-point, 2-points crossover and uniform crossover and found that it performs better than both in terms of precision where it achieves precision of 0.4464 at P@10, using CACM document set which includes 3204 documents, and it is better than uniform crossover by 35.36%.

2.4.7.4 Assessment of Crossover Techniques

The main feature that must characterize the crossover operator is to generate optimal chromosomes by inheriting good features from parents and maintains the chromosome performance at its highest value. Therefore, the crossover technique must be implemented in a manner that passes good genes from one generation to another up to the last generation and to minimize the loss of these good genes during generation creation. The main drawback of the previously discussed crossover techniques is that the generated offspring may be of lower performance than their parents such in case of n -point crossover where n is greater than zero. It is proved when the technique applied by (Vrajitoru, 1998), where the average precision-recall scored ranges between 0.2182 and 0.4149. On the other hand, (Klabbankoh and Pinngern, 2008) achieved recall as high as 0.976 and precision as high as 0.746. Although these results are promising, but these are affected by the document model and the fitness function applied.

However if the one-point crossover method is combined with another crossover method such as re-ordering crossover then there is a great opportunity to have offspring with a better performance as expected by (Goldberg and Bridges, 1990) and proved in chapter 4 experimentally. Another drawback of the some crossover techniques is that they require extra time when re-producing the offspring such as reordering crossover and arithmetical operational crossover. The third drawback in some of the above crossover techniques is that they require additional storage space such as inversion crossover method. Destroying building blocks of consecutive genes that perform better together is a common drawback

for many crossover techniques such as in uniform crossover, random crossover and n -point crossover when n is relatively large. For the uniform crossover, it is shown that when applied by (Xu, Deli, and Yu, 2009), the achievement was only 0.75 for the precision-recall@10, which is comparatively low. When applying the n -point crossover by (Klabbankoh and Pinngern, 2008), the maximum precision achieved is 0.746. and the lowest is 0.417. However, the last approach was applied in GA using binary representation, and there is a need to investigate its performance when applied on other kind of representation. The last drawback that needs to be mentioned here is that the generation of the offspring may produce an individual which is out of the search space such as in case of arithmetic crossover which also increases resource requirements in terms of time and processing. An empirical study is performed in chapter 4 to compare the performance of one-point, two-point, ordered crossover and fusion crossover in terms of recall and precision for inter representation. Summary of crossover techniques along with the advantages and disadvantages are listed in Table 2-4.

The above discussion emphasizes the need to develop a new crossover technique or enhance the existing crossover operator such that it overcomes the above mentioned drawbacks and to be applied with probability equal to one, in contradiction to the most of researchers where they apply it with probability between 0.6 and 0.8 as explained earlier.

Table 2-4: Summary of crossover techniques used in GA

Crossover Method	Description	Advantages	Disadvantages
Single point crossover (Lopez-Pujalte, Guerrero-Bote, and de Moya-Anegon, a, 2003; Song and Park, Asllaniand and Lari, 2007; Radwan et al 2006; Aly, 2007; Yeh et al 2007; Húsek et al, 2005; Billhardt et al, 2002; Vrajitoru, 2000; Vrajitoru, 1998; Beasley, Bull, and Martin, 1993a; Desjardins, Godin, and Proulx, 2005; Martín-Bautista and Vila, 1998; Klabbankoh and Pinngern, 2008; Carroll	Randomly choose a point and genes of parents are exchanged before or after this point.	<ul style="list-style-type: none"> • Easy to implement • Fast in generating child 	<ul style="list-style-type: none"> • Best building blocks can be broken. • offspring may have lower performance than parents • The offspring can be similar to parent if crosspoint is close to one edge of the parent • may produce child with lower performance than parent

and Lee, 2008; Simon, Sathya, 2009; Drias, Khennak, and Boukhedra, 2009;)			
Two-points crossover (Pathak, Gordon and Fan, 2000; Beasley, Bull, and Martin, b, 1993; Yang, Korfhage, and Rasmussen, 1992; Spears and De Jong, 1991; Atsumi, 1997)	Randomly choose two points and genes of parents between these 2 points are exchanged.	<ul style="list-style-type: none"> • Provides wider diversity from one-point crossover • Performs better than n-point crossover (Spears and De Jong, 1991). 	<ul style="list-style-type: none"> • The offspring can be similar to parent if crosspoints are close to each other • performance using 2-point crossover drops dramatically if the recommendation of building blocks are not adhered to (Beasley et al,1993b)
n-point crossover (Vrajitoru, 1998; Klabbankoh and Pinngern, 2008; Kazarlis, Papadakis, Theocharis and Petridis, 2001; Spears and De Jong, 1991)	Randomly choose a number of crosspoints and apply n simple crossover operations on the parents at once	<ul style="list-style-type: none"> • Provides wider diversity from two-points crossover 	<ul style="list-style-type: none"> • Reduce speed of the crossover process • Disrupts the building blocks • may produce child with lower performance than parent
Uniform crossover (Cutler et al, 1999; Beasley, Bull, and Martin, b, 1993; Yang, Korfhage, and Rasmussen, 1992; Muhlenbein and Schlierkamp-Voosen, 1993; Xu, Deli, and Yu, 2009)	Random binary mask is generated to swap corresponding 1's positions or preset probability for swapping each gene	<ul style="list-style-type: none"> • offspring have high chance to differ from parents if swap probability is high • finding optimal solution faster • order is not important • increased disruption is beneficial if the population size is small produced robust performance (Beasley et al,1993b) 	<ul style="list-style-type: none"> • Disrupts the building blocks. • may produce child with lower performance than parent
Random crossover (Alfonseca, 1991).	exchange randomly selected set of genes between two parents	<ul style="list-style-type: none"> • Wide range of search space to select from. 	<ul style="list-style-type: none"> • Disrupts the building blocks. • may produce child with lower performance than parent
Recint crossover (Minaei-Bidgoli and Punch, 2003).	$Offspring = parent1 + Alpha \times (parent2 - parent1)$	<ul style="list-style-type: none"> • No justified advantage 	<ul style="list-style-type: none"> • Applicable on real numbers only. • High probability of offspring to be similar to parents • generated offspring could be out of the search space
Arithmetical operation (Kim and Zhang, 2003; S. Kim, B.-T. Zhang, 2000).	offspring is average of parent's genes	<ul style="list-style-type: none"> • Allow to generate new child which are not exist in the original space. 	<ul style="list-style-type: none"> • Generated offspring could be similar to parents • week solution by diverting from the optimal solution • generated offspring could be out of the search space • Requires additional

			processing time in order to find the average.
Inversion crossover (Beasley et al, 1993b).	Reverse the order of genes between 2 crosspoints	<ul style="list-style-type: none"> • It helps finding gene orderings which have better evolutionary potential 	<ul style="list-style-type: none"> • Child may have lower performance than parents. • Genes must be labelled in this method if maintaining same position within the chromosome which requires additional storage (Beasley et al, 1993b). • Requires additional processing time in order to order the genes.
Reordering crossover (Beasley, Bull, and Martin, 1993b; Yang, Korfhage, and Rasmussen, 1992)	orders the genes between 2 randomly chosen positions within the chromosome either in ascending order or descending order	<ul style="list-style-type: none"> • useful when needing to discover good gene ordering 	<ul style="list-style-type: none"> • Reduce speed of the crossover process for ordering each chromosome.
Fusion crossover (Vrajitoru, 1998). Produces.	Produce only one child from two parents where for each gene, the child inherits the value from one or the other of the parents with a probability according to its performance	<ul style="list-style-type: none"> • good genes of both parents are inherited simultaneously • speeds up finding the optimal solution 	<ul style="list-style-type: none"> • Reduce speed of the crossover process
Dissociated crossover (Vrajitoru, 1998; Collard and Escazut, 1995; Yan et al, 2009).	Genes in second child are replaced by 0 between the cross points in binary representation or replaced by the function: Offspring = parent1 + Alpha × (parent2 – parent1). Alpha is a Scaling factor chosen uniformly in the interval [-0.25, 1.25].	<ul style="list-style-type: none"> • Performs better in terms of precision when compared with classical 1-point, 2-points crossover and uniform crossover (Collard and Escazut, 1995) 	<ul style="list-style-type: none"> • Generated offspring could be out of the search space
Swap-Window (Petridis, Kazarlis and Bakirtzis, 1998).	Selects two arbitrary unit strings u1, u2, a “time window” of random width and a random window position taking values in the range . Then the bits of the two unit strings u1,	<ul style="list-style-type: none"> • Could enhance gene performance if used as secondary operator in addition to main crossover technique. 	<ul style="list-style-type: none"> • Window size could be small which doesn't allow for enhancement of the offspring over the parent.

	u2 within the window are interchanged		
--	---------------------------------------	--	--

2.4.8 Mutation Operator

Mutation is the third and last genetic operator. It causes the individual genetic representation to be changed according to some probability p_m ranging from 0.001 (Beasley et al, 1993a) to 0.7 (Radwan et al 2006). Mutation is generally considered to be a back-ground operator since it ensures that the probability of searching a particular subspace of the problem space is never zero (Minaei-Bidgoli and Punch, 2003). Despite it is applied with low probability, it is considered as a very important operator (Beasley et al, 1993b). Applying mutation during GA achieves many objectives. These objectives are restoring lost data, exploring variety of data (Radwan et al 2006), improving diversity of the solution (Noreault et al, 1980), and reduce the possibility of converging to a local optimum, rather than the global optimum (Minaei-Bidgoli and Punch, 2003; Noreault et al, 1980).

The most common method of mutation is implemented by randomly selecting one gene and changing it with another value (Radwan et al 2006; Aly, 2007; Alfonso, 1991; Vrajitoru, 2000; Vrajitoru, 1998; Beasley et al, 1993a; Martín-Bautista and Vila, 1998; Klabbankoh and Pinngern, 2008; Lopez-Pujalte, Guerrero-Bote, and de Moya-Anegon, a, 2003). Example of random mutation is shown in Figure 2-8, where position 8 is selected randomly and its value is replaced by another random value.

$$C_i = \{2, 3, 4, 5, 7, 9, 11, \mathbf{12}, 22, 23\}$$

$$C_i = \{2, 3, 4, 5, 7, 9, 11, \mathbf{15}, 22, 23\}$$

Figure 2-8: Example of random mutation

Asllani and Lari (2003) suggested mutation by randomly selecting two genes within a chromosome and swap their positions. This method is useful if the order of genes within the chromosome is important. Mutation can also be performed by introducing Gaussian noise (Pathak, Gordon and Fan, 2000; Song and Park, 2009; Steinbach, Karypis, and Kumar, 2000). While previous methods apply mutation to particular genes, it is found that Beasley, Bull, and Martin (1993a) apply mutation for each gene but with probability

of 0.001. This is to ensure that no point in the search space has a zero probability of being examined as per the authors. When genes are presented in a numerical form (Beasley et al, 1993b) then mutation is performed in one of three methods. Either by replacing the value with a random one, or using Creep method which is adding or subtracting a small randomly generated amount, or using geometric Creep method which is multiplying by a random amount close to one. In the work done in (Yang, Korfhage, and Rasmussen, 1992) some individuals are selected randomly from the semi-new generation, and some of their genes are selected and assigned new values, also in the range [0, 1]. The number of individuals to be selected depends on the mutation ratio chosen in the experimentation. The selection of genes and the new value assignments are also done randomly. A new mutation method introduced by Horng and Yeh (2000) was applied to generated offspring. It is defined to be the inversion of a weight as per the following formula:

$$W_i^{\text{old}} - W_i^{\text{best}} = W_i^{\text{best}} - W_i^{\text{new}},$$

where W_i^{old} is a weight of a gene in the population; W_i^{best} is the weight of the best gene in the population, and W_i^{new} is the weight of a gene to be found in the new population.

Last method of mutation to be mentioned in this literature is the one suggested by Wang and Feng (2005) to speed up the ability of finding better query vector. In this method the new individual which represents a query in this model is assigned the term weight average of selected individual query if $\text{random}(p) < p(\text{mutation})$, otherwise it is assigned a maximum term weight of selected individual minus the minimum term weight of the selected individual. (Yan et al, 2009) use quantum-bit representation for the gene which is defined as pair of numbers (α, β) as: $\begin{pmatrix} \alpha \\ \beta \end{pmatrix}$. Mutation in this technique is performed by swapping the values of α and β for the mutated gene.

2.4.9 Assessment of Mutation Operators

As mentioned earlier, some of these techniques cannot be applied to our model as the resulted genes may not belong to the search space like in arithmetic operation mutation. It

cannot also be applied when assigning random value to a gene where this value could not be in the search space. Moreover, the mutation could destroy the building blocks if the mutated gene has lower performance than the replaced one. In these cases if a validation is done before replacement to make sure that the replacing gene is of same or better performance (fitness value) than the replaced one, then these drawbacks could be avoided. Another drawback is the slow performance of mutation when performing some operations to produce new genes via mutation such as the arithmetic operation or when introducing Gaussian noise. Finally, in case of Q-bit gene representation, the swapping of α and β for the mutated gene may produce a duplicate gene, consequently, the performance of the new chromosome is not enhanced. As a result, the mutation could be applied to one or two individuals within the chromosome ensuring that the replacing gene enhances the overall performance of the chromosome. Table 2-5 summarizes the mutation methods with their advantages and disadvantages.

Table 2-5: Summary of mutation methods used in GA

Mutation method	Description	Advantages	Disadvantages
One position mutation (Radwan et al 2006; Aly, 2007; Alfonseca, 1991; Vrajitoru, 2000; Vrajitoru, 1998; Beasley et al, 1993a; Martín-Bautista and Vila, 1998; Klabbankoh and Pinngern, 2008; Lopez-Pujalte, Guerrero-Bote, and de Moya-Anegon, a, 2003; Carroll and Lee, 2008; Saini, Sharma, and Gupta, 2011)	Replace random position by random value	<ul style="list-style-type: none"> Introduces new gene in the chromosome if duplicates are discarded 	<ul style="list-style-type: none"> Mutated gene can be of lower performance than before mutation.
Two position mutation (A. Asllani and A. Lari, 2007)	Selecting two randomly genes within a chromosome and swap their positions	<ul style="list-style-type: none"> Useful only if order of genes is important 	<ul style="list-style-type: none"> useless if order not important
Introducing Gaussian noise (Pathak, Gordon)	One location selected randomly and replaced with Gaussian noise	<ul style="list-style-type: none"> Introduces new gene in the chromosome 	<ul style="list-style-type: none"> Extra calculation load on the system

and Fan, 2000; Song and Park, 2009; Steinbach, Karypis, and Kumar, 2000)			
Probability mutation (Beasley et al, 1993a)	Applies mutation to each gene but with probability	<ul style="list-style-type: none"> • Provide wider diversity of genes 	<ul style="list-style-type: none"> • Damage building blocks • slow generation process
Arithmetic operation mutation (Beasley, Bull, and Martin, 1993b)	Add or subtract a small, randomly generated amount or multiply the gene value by a random amount close to one	<ul style="list-style-type: none"> • Introduces new gene in the chromosome 	<ul style="list-style-type: none"> • Could produce duplicate genes
Assign random value between 0 and 1 to some randomly selected individuals (Yang, Korfhage, and Rasmussen, 1992)	Assign random value between 0 and 1 to some randomly selected individuals	<ul style="list-style-type: none"> • Introduces new gene in the chromosome if duplicates are discarded 	<ul style="list-style-type: none"> • Applicable for genes ranging between 0 and one only. • Could produce duplicate gene
Inversion of a weight of mutated gene (Horng and Yeh, 2000)	$W_i^{old} - W_i^{best} = W_i^{best} - W_i^{new}$	<ul style="list-style-type: none"> • Introduces new gene in the chromosome 	<ul style="list-style-type: none"> • Requires knowledge of all genes in the population in advance.
Swapping the values of α and β for the mutated gene in Q-bit representation (Yan et al , 2009)	Swapping the values of α and β for the mutated gene in Q-bit representation	<ul style="list-style-type: none"> • Introduces new gene in the chromosome 	<ul style="list-style-type: none"> • The produced gene may be a duplicate of existing gene.

2.5 Summary

The following observations are concluded from the literature review of the IR categories in general and the literature review of the genetic algorithm technique and its operators in particular. First observation is that the traditional IR models are not practical to be applied to the web search domain due its complexity and the expensive computation cost. The second observation is that there is no research work–up to our knowledge- that analyzed all GA operators in one study and no one –also up to our knowledge- tried to find the best combination of GA operators that produce the highest performance of information retrieval systems in terms of recall and precision. The third observed point is that the GA operators in the reviewed studies are tested and analyzed on documents represented by either vector space model or latent semantic model while in this research

they are tested and analyzed using modified inverted index model. The fourth important point is that most of the studies reported in the literature are using ready-made indexes. The fifth observation is that the precision and recall achieved by these techniques and models are still not meeting the user satisfaction which is the major aim of any information retrieval system, and this motivates the researchers to keep on developing several techniques of IR.

It is noted in this literature review that the average precision ranges between 0.2182 (Vrajitoru, 1998) and 0.4149 (Vrajitoru, 1998), and the average precision for recall ranges between 0.225 (Desjardins, Godin, and Proulx, 2005) and 0.7003 (Horng and Yeh, 2000) using the 11-points average while it is reaching 0.2969 using the 9-points average. For recall at N it is ranging between 0.274 and 0.319.

Although several techniques are developed for each operator of GA, some are not suitable for the proposed IR model, while others are producing good results based on some environmental setting, but still the achieved precision and recall percentage are low compared to the user expectation. On the other hand, some GA-based IR models produced high percentage of precision, but this was due to the relevance feedback provided by the end user which is not considered in this study.

In the light of these results and observations, there is an important need to have an improved IR system that provides high recall and high precision results. Consequently, this thesis presents two models of IR system that aim at filling in the gap of the low performance of IR systems in terms of recall and precision. First model is GA-based IR model while the second is traditional IR-based model. Details of these models, their features and their performance are presented in the next chapters.

C Chapter Three: The Design of Information Retrieval Using Genetic Algorithm (IRUGA) Model

3.1 Introduction

Recently, IR problems have gained a considerable importance, and most studies argue that IR can be seen as a standard optimization problem (Marghny and Ali, 2005; Petridis, Kazarlis and Bakirtzis, 1998; Deb, 1998). Therefore, many researches are directed towards the use of Genetic Algorithms (GAs) for developing such a system which has proved its simplicity and capability as a powerful search mechanism to solve many scientific and engineering problems (Minaei-Bidgoli and Punch, 2003; Asllaniand and Lari, 2007; Losee, 1996; Deb, 1998). As is clear from its title, the goal of this thesis is to utilize the concept of GA with a significant improvement to produce what is called: “Information Retrieval Using the Genetic Algorithm (IRUGA) model”.

IRUGA consists of two main units. The main purpose of the first unit, namely indexing, is to extract the meaningful keywords from the documents and represent them in a way that makes the process of finding relevant documents efficient. GA is the second unit of IRUGA and is utilized in this thesis as a core of its behaviour. This unit compares the user query with the indexed documents to retrieve the relevant set of documents and display them in a descending order according to a relevance measure.

More precisely, in order to obtain high quality results, additional units need to be combined with IRUGA, namely, the query formatting unit and the ranking unit. These units are outlined as shown in Figure 3-1.

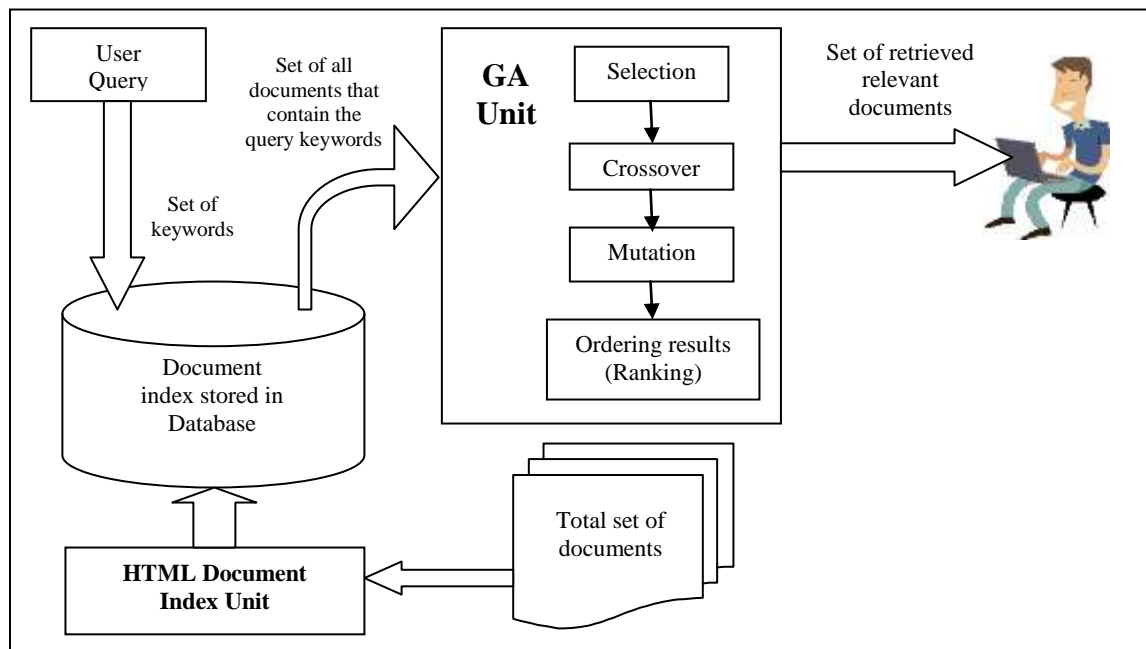


Figure 3-1: The units of IRUGA

The rest of this chapter is organized as follows: Section 3.2 highlights some of the document types as well as the indexing engine even though one of them will be selected throughout the IRUGA's designing. The goal of this section is to talk about the reasons for using HTML documents being used as a backbone for IRUGA, and the HTML tags that are to be used to classify words within the document and finally the assigned weights of these tags, while the second part of this section explains the new approach of the document indexing which is designed especially for IRUGA. Moreover, this section is divided into three parts. The first one is the reasons for using the inverted index model in IRUGA, while the second part describes how the data is stored in this index, and finally the mechanism for creating the index is presented at the end of this section. Section 3.3 describes in full details the main unit involved in IRUGA, that is GA. New techniques are explained and applied to the GA of IRUGA and to both new functions which are named as: multi-terminal function and term proximity functions are also explained in this section. The last section of this chapter, which is 3.4, summaries this chapter.

3.2 Document Types and Representation

The effective way to represent documents influenced scientists' thought in the IR arena. Historically, documents that are evaluated by IR can be either plain text, semi-structured (i.e., HTML (Hypertext Markup Language) documents) or structured. Because most of web-documents are written in HTML (Kim and Zhang, 2000; 2003), this format was adopted for implementing our proposed system. In addition to that, many GA-based IR systems are implemented using plain text such as (Radwan et al, 2006; Vrajitoru, 1997; Billhardt et al, 2002; Cummins and O'Riordan, 2006; Aly, 2007; Salton and Buckley, 1990) which do not comply with the Web search systems.

3.2.1 Document Type to be used by IRUGA

Actually, the documents that are indexed by the inverted index are the semi-structured (i.e.: HTML (Hypertext Markup Language)) documents. This format exhibits the following features:

1. IRUGA is an application of web mining where most of the web documents are written in HTML format (Kim and Zhang, 2003; Kim and Zhang, 2000).
2. The layout and HTML tags better reflect the importance of certain terms within a document (Cutler et al, 1999) where the emphasized terms are written in a different format.
3. The ability to use HTML tags in weighting the terms (Cutler, Shih and Meng, 1997) as will be illustrated in the next sub-section.
4. HTML documents are well content descriptive documents (Cutler, Shih and Meng, 1997).
5. Clarity of the tags that need to be included in the term weights (Kim and Zhang, 2003; Cutler et al, 1999), where these tags are bounded by special characters to distinguish them from the normal text within the document.

Because of these features, HTML documents are adopted to form the search space of IRUGA.

3.2.2 HTML Tags and Weights Used in IRUGA

The main characteristic of this type of documents is the HTML tags. These tags reflect different kinds of importance for each portion of the document. Basically, these tags can be classified into semantic tags and formatting tags. Semantic tags emphasize the ideas that the author needs to highlight, while formatting tags are related to the way of writing the syntax. Examples of the former are the *title*, *header* (in all levels: header1, header2, header3, ... header6), *anchor text*, and *body*. These tags are represented as TITLE, HEADER, H1, H2, H3, A, and BODY, respectively, while examples of formatting tags are *bold* and *italic* tags and they are represented by *B* and *I* tags. Actually, each tag within the document reflects a certain level of importance. However, IRUGA will be using only these tags as they better reflect the document content in terms of user query. Moreover, they are most frequently used in the web documents (Cutler et al, 1999). Example of HTML document is illustrated in Figure 3-2 where the HTML are shaded.

```

<html>
<head>
<title>Donald P. Greenberg
</title>
</head>
<body>
src="http://www.cs.cornell.edu/Info/People/greenberg/greenberg-thumb.gif"
<h2>
Donald P. Greenberg <br>
Jacob Gould Schurman Prof. of Computer Science <br>
Director, NSF Science and Technology Center for Computer Graphics and
Scientific Visualization <br>
PhD Cornell University, 1968
</h2>
For the past two decades, our computer graphics activities have involved
the development of a wide range of graphic input and display techniques. A
number of input methods have been implemented, and progress has been made
on a large variety of display routines. Graphics research topics
previously investigated include polygon clipping, hidden surface
algorithms, texturing, spatial and temporal aliasing problems, geometric
modeling, parametric surface descriptions, and color science.
<p>
<H2>Professional Activities</H2>
<ul>
<li>Fellow, ACM
</ul>
</body>
</html>

```

Figure 3-2: Sample of HTML document

To better understand the importance of these tags, an explanation of each one of these tags is provided here, starting with the most important tag, which is the title tag. The terms in the document's title provide information about what the document is about. Hence, a score of 6 is given to the terms that occur in this tag. The second more important tag is the header tag and its sub-headers, denoted by H1, H2 and H3. The words in these tags provide information about the structure and main topics of the document; hence, a score of 5 is assigned to the terms within this tag. The next important HTML tag is the anchor tag. The words in the anchor appear in the hyperlink which points to other documents. These documents which are pointed to by the hyperlink are most likely to be relevant to user query. In addition, the anchor tag contains words that appear in other documents. Therefore, terms within this tag are given a weight of 3. Authors emphasize certain terms within the HTML document by using specific formats such as bold and italic to reflect a degree of importance that they want to highlight; hence, this tag provides an extra level of describing document content. For these reasons, these two tags are given a weight of 2. The last tag to be mentioned here is the body tag. This tag contains plain text. This plain text cannot be neglected as it plays a limited role in identifying the document. Therefore the words in the body text are given a weight of one. These weights are influenced from the study done by Kim and Zhang (2003) and summarized in Table 3-1.

Table 3-1: The weight assigned to HTML tags used in the inverted index

Tag Name	Weight
Title	6
Head, h1, h2, h3	5
A: Anchor	4
B: Bold, I: italic	3
Body	1

Terms which are used in formatting the document or terms that appear in the document and have nothing to do with document content are excluded.

3.3 Inverted Index Unit

As mentioned in the introduction of this chapter, the process of indexing documents is considered as one of the main units of IRUGA. It is designed to represent documents that are used to support the GA unit. This representation is done through a process called

enhanced inverted index. As shown in Figure 3-1, the document set is passed to the indexing unit. Then this unit extract the useful information from these documents and store it in the database for later reference by the GA unit.

Definition 3.1: Inverted index: is a structure that attaches each distinctive term to a list of all documents that contain the term (Liu, 2006, p. 205).

3.3.1 Why the Inverted Index Model

It is proved that effective representation in the area of IR is achieved by selecting a proper index terms (Pathak, Gordon and Fan, 2000) and choosing a proper indexing model (Radwan et al 2006). Inspired by several researches, (described in Section 2.2.2.2), IR systems that adopt GA as a backbone of their systems are using vector space model to represent the documents (Lopez-Pujalte et al, 2003a; Aly, 2007; Vrajitoru, 1997; Billhardt et al, 2002). On the other hand, several scientists are using Latent Symantec model to develop their indexing scheme (Kleinberg and Tomkins, 1999; Song and Park, 2009). These models may not be used any more in the areas of IR for the following reasons (Snasel et al, 2005):

1. They require a large amount of space to store the index.
2. They require a long time to retrieve the needed keyword.
3. The need to examine all the documents against each query.
4. The index stores limited information about the documents (Al-Dallal and Abdulwahab, 2011), and this is due to the large amount of space required to include addition data per keyword.

To overcome these drawbacks, a well known indexing scheme is developed and was chosen for the implementation of our system. This indexing scheme is called the inverted index (Liu, 2006). It is perhaps the most important index method used in search engines as stated by Liu (2006, p. 204).

Furthermore traditional IR seeks to find documents that match the keyword of the user query. To do so, all documents are scanned sequentially in the database to find what the IR is seeking for. However, this procedure is impractical for large collections of documents

such as the Web. Another option which is more efficient in terms of speed and processing is to scan the database for the query keywords and retrieve the documents that index these keywords.

This thesis will concentrate on developing the new indexing scheme to overcome these drawbacks. This indexing is developed by improving the known inverted index. Generally speaking, this model of indexing scheme is characterized by a set of the following features:

1. The building time of indexing documents is short and the performance of parsing the document set cannot be simply ignored. Indeed, the time complexity required to build this index is $O(n)$ where n the number of all terms in the document collection (Liu, 2006, p. 204).
2. Liu (2006) presented good remarks regarding the behaviour of this model for its ability to retrieve variant documents that matched the terms of the user query.
3. Liu (2006) also pointed out that it is possible to identify documents by using the pointers inside the inverted list, so no search operation is carried out on the documents themselves.
4. The space required to store the result of this model is very small compared to the space required by other models (Snasel, Moravec and Pokorny, 2005). Indeed, only the indexed terms and the referenced documents will be saved.
5. Different types of information related to each term such as frequency, position and weight can be easily stored. Thus, the evaluation time of the term is reduced dramatically.
6. It is obvious that this model is very fast in retrieving the documents as it scans the database for the query keywords and retrieves the documents that index these keywords, instead of examining each document against the query (Uematsu et al, 2008).

In conclusion, one can easily see that the performance of this model has been emphasized by many researchers (Liu, 2006; Uematsu et al, 2008). For this reason, the inverted index model is adopted in IRUGA with a significant improvement to produce what is called the

Enhanced Inverted Index (EII).

As mentioned earlier in this section, the concern when building the index is the choice of the model to be used and the terms to be indexed. Now the talk is about the second part which identifies the terms that need to be indexed.

It is obvious that any index must contain all information that allows the best retrieval for the required document based on keywords provided by the end user. Part of this information is obtained while parsing the document during the process of creating the index such as: the referencing documents, the offset (position) within the document, the offset within the sentence and word weight based on the HTML tag that it is bounded with. On the other hand, there is information obtained once the index creation is over. This is classified as global information. Global information includes the total number of documents in the collection, total number of indexed terms, total number of unique terms in the collection, and total weight of all indexed terms (Cummins and O’Riordan, 2006). All these types of information need to be stored in the index in order to facilitate the process of evaluating the document based on the user query and consequently to retrieve the proper documents.

Nevertheless, there is a collection of words which don’t have much informative content. This type of words is not useful in identifying the document; hence, they are omitted from the index. These words are known as stop-words. The storage space could be reduced by 30% when excluding stop-words from the index (ixCreateStopWordList, 2002). There are many lists of stop-words available in the Web but there is no reason for favouring one of them over the rest. Hence, the list of stop-words used when creating EII is the one provided in (Stop Word List 1, 2002).

3.3.2 Inverted Index Mechanism

The inverted index is one of the most interesting models used for indexing the documents. The idea of the inverted index as described in (Uematsu et al, 2008) is a structure used to store word position data, as well as document ID. Word position data is a list of offsets or positions in which the words occur in the document. Such occurrence information (i.e. document ID and word position data) for each word is expressed as a list, called the “inverted list”, and all the inverted lists taken together are referred to as the inverted index.

The position data is mainly used when the order of words within the query is sensitive and also used for proximity search when the distance between words is required. When a query phrase is submitted, the EII unit accesses the inverted list of each word of the query to identify the documents that contain those words in the same order as the query if possible. In addition, EII retrieves the additional information associated with the query words such as the offset to be fed into the evaluation process, which is the fitness function.

However, as stated above, the inverted index has motivated many developers to design their own schemes. This section highlights some of the most prominent improvements and variations of the inverted index scheme. Here, the inverted index has been modified by associating each word with its weight and its position within the sentence and within the document.

In particular, two different mechanisms are applied to implement the inverted index in IRUGA. The first mechanism is implemented by using the link list data structure through using C++ programming language. Furthermore, Oracle database with its tables and relations is used to implement the second mechanism. In fact, both mechanisms were examined in this work and their details are described in the following sections.

3.3.2.1 Building the Inverted Index Using C++ Data Structure

The first adopted option was to implement this system using C++ programming language. The basic concept in this method is the link list data structure. In its simplest form the inverted index of a document collection is basically a data structure that attaches each distinctive term to a list of all documents that contain the term. Therefore, the system is built using two types of nodes where each type is linked to its peers through a link list structure. The first type of nodes is the *word node*. It includes the word to be indexed, the total frequency of this word in the document set and the total number of documents referencing this word. The second type of nodes is the *document node*. It is linked to the word node and stores:

- a. Document name.
- b. Total number of words in the document.
- c. Frequency of that word within the document.

- d. The HTML tag weight of this word in the document.

There can be as many nodes linked to the word node as there are documents referencing this word.

The structures of the *word node* and *document node* are shown in Figures 3-3 and 3-4 respectively, while Figure 3-5 shows how all nodes are linked together.

This model of implementing the inverted index encounter two main drawbacks during the implementation; the first one is that it requires a lot of memory to build the index, causing the computer to experience undesired slowness. The second drawback is the slowness of retrieving details of a query word. The slowness comes from the fact that accessing a word node from the list requires parsing the word list sequentially until it reaches the required word node.

In order to overcome these drawbacks, another method of implementing the inverted index is used; in this method, Oracle database is used instead of the link list data-structure, and PL/SQL programming language instead of C++. The algorithm of building the inverted index using C++ data structure is listed in Algorithm 3-1.

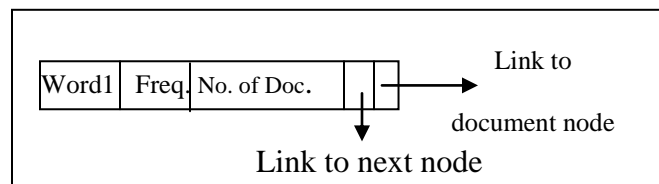


Figure 3-3: Word node

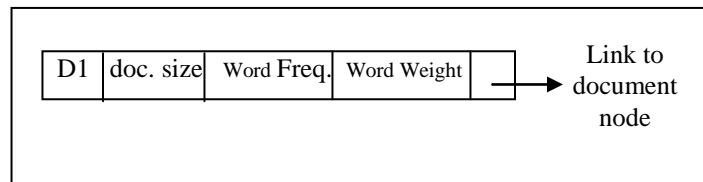


Figure 3-4: Document node.

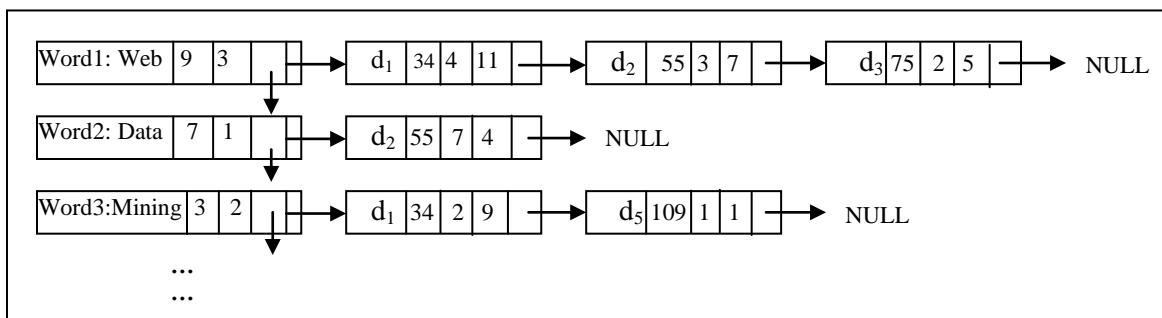


Figure 3-5: Data structure constructed by the inverted index showing how word node is linked to a list of document nodes.

3.3.2.2 Building the Inverted Index Using Oracle Database

In this model, several tables are used to represent different objects of the inverted index model. The parser engine is driven by table *file_info* which includes names and references of the documents that need to be indexed. Two setup tables are used. One is called *tag_weight* which stores the HTML tag and its corresponding weight, and a second is called the *special_words* table, which stores the stop words, special characters and sentence

Algorithm 3-1: Indexing engine using C++ data structure

```

while there are more documents that need to be processed do
  while there is a word in the document do
    Read the document one word at a time until the whole document is read. Split the string into
    tokens.
    Remove stop words
    Get the related document node, or create it if it doesn't exist
    if there is a link to document node for same document
      then
        increment frequency in that node
      else
        create new node for this document and set frequency to 1
        Add this document to the list of documents
      end if
    Increment the word count for that document since it is needed in calculating word density
    per document
  end while
end while

```

delimiters. Stop words are the words that occur very frequently in the text but have little meaning (Liu, 2006). Special characters are set of characters to be discarded while parsing the files, and which are used as punctuation or formatting. Sentence delimiters are a subset of special characters that are used to separate the sentences from each other, so the words belonging to each sentence have the offset of the sentence or the sentence number. These delimiters include period, semi colon, question mark and exclamation mark.

The process of the inverted index starts by reading the document name from the table

file_info, opens that document, reads it word by word and ignores the stop words. In addition, the scanning process will ignore the HTML tag words, formatting words, and the words that are outside a tag since these words could be used for formatting the document or may be hidden. The inverted index process classifies the words depending on the tag that occurs within, and accordingly assigns a weight to each word as per Table 3-1 (page 67).

EII computes *word_weight* variable for each document which is considered one of the computed global variables via this algorithm. Initially, at the time of reading the document, the variable *word_weight* is set to zero. The *word_weight* is incremented by the HTML tag weight fetched from the *tag_weight* table. Since the word can be nested in HTML tags (i.e. within several tags at the same time so as to be in the header and written in bold and italic font), then it will have the accumulation of these tag weights (e.g.: 5+3+3=11 for header, bold and italic tags as per Table 3-1).

A new record *R* will be added to table *word_doc* if the intended word is new within the document. EII adds other variables to *R*, i.e., $R = [word, document, frequency, term\ weight]$ where *R.frequency* is set to 1 and *R.term_weight* is equal to the value of the *word_weight*. On the other hand, if the combination of the [word, document] already exists then its frequency is incremented by one and the global variable *word_weight* is incremented by *tag_weight*. If the closing HTML tag is found, then the global variable *word_weight* is decremented by HTML tag weight fetched from the *tag_weight* table. In addition to the *word_weight*, the positions within the sentence *ps* and within the document *pd* are included in the *word_doc* table. Algorithm 3-2 shows the pseudo code for the parser engine.

Algorithm 3-2: The document set parser

```

While  $d \in D$  do
  global_term_weight = 0;
  While not EOF do
    ps = 0;
    pd = 0;
    Get word (w1);
    If w1 is an open tag
      global_term_weight := global_term_weight + tag_weight;
    Else if w1 is a closing tag
      global_term_weight := global_term_weight - tag_weight.
    Else if w1 is not stopword
      {
        w1.weight = global_term_weight;

```

```

If  $w1 \notin keyword$  table then insert it with frequency =1
Else  $keyword.w1.frequency = keyword.w1.frequency + 1;$ 
If this is first occurrence of the word in the document then
     $unique\_terms = unique\_terms + 1;$ 
    Insert new record R in table  $word\_doc$  with values [document name, w1, frequency
    of 1,  $global\_term\_Weight$ ]
    Insert new record in table  $word\_position$  with values [w1,  $ps$ ,  $pd$ ].
Else  $w1.frequency = w1.frequency + 1;$ 
     $word\_doc.w1.term\_weight = global\_term\_weight + word\_doc.w1.term\_weight$ 
     $file\_info.file\_size = file\_info.file\_size + 1;$ 
}
End while.
End while.

```

By the end of the parsing process, the index details are included in three tables. The first one is the *word_doc* table which stores frequency, total weight of the word within the document, and the documents that reference each word. The second table is *word_position* which stores the offset (position) within the document and the sentence number for each word. The last one is the driver table *file_info*, which includes global details for each document. These details are: document size, summation of words' weight within the document, and the total number of unique words.

Recall that in Algorithm 3-2, the index details are appended to three tables. The details of these three tables are shown below:

1. Each record in the *word_doc* table (*TW*) is a 3-triple [F, W, P], where F is used to store the frequency of the word within the document, W is the weight, and P is the document name that references this word.
2. Each word of the document will have a set of records in the *word_position* table (*TP*) where each record is also a 3-triple [T, ps, wd], where T is the word (Term), while ps and wd are as described earlier.
3. Each record in the table *file_info* is a vector including global details for each document in the form of [$DS, WT, UT,$] where DS is the document size, WT is the summation of words weight within the document, and UT is the total number of unique words.

Once the documents are parsed, the inverted index is created and the pre-processing step is

over; so that all documents are presented in a way that IRUGA can process these documents in order to retrieve the related documents and evaluate them based on the user query. Moreover, the data is available for document evaluation based on user query. Figure 3-6 shows the interaction between the EII engine, the input tables that supply the parser engine with the required data and the output tables that store the output of the parser engine. The Flowchart of creating the EII is demonstrated in Figure 3-7.

Once the index is created, the pre-processing step is over and the IRUGA is ready to start receiving queries.

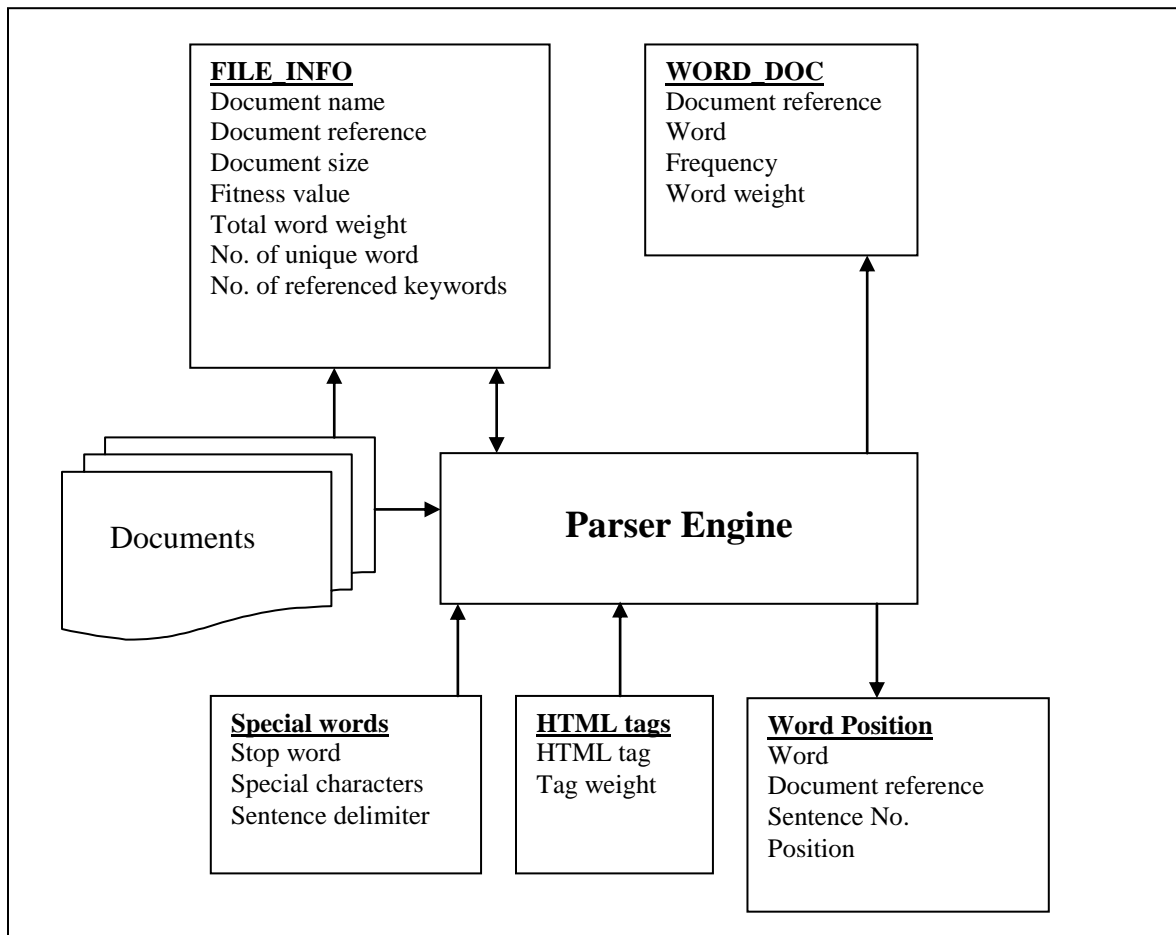


Figure 3-6: The parser engine showing the input tables and the output tables

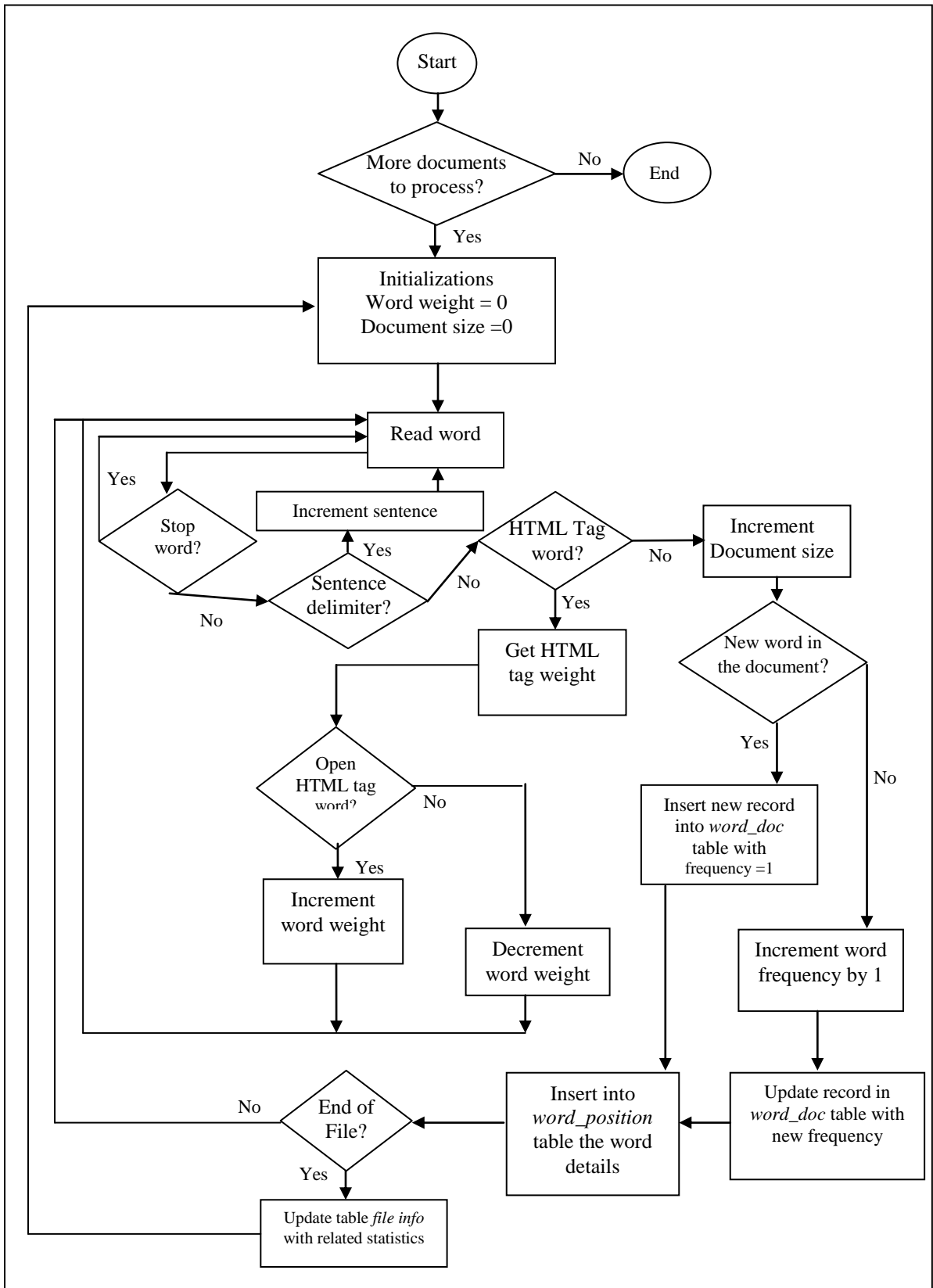


Figure 3-7: The flowchart of creating the inverted index.

3.4 GA Unit:

As mentioned in the introduction, the IRUGA consists of two main components: the indexing unit and the GA unit. In this section, the new implementation of IRUGA's GA mechanism will be explained. The reason behind choosing GA to be the backbone of IRUGA is that it has become a popular technique used in IR where many approaches are developed based on GA. This is due to the following features of GA:

1. It is a stochastic algorithm where randomness is an essential role in genetic algorithms as two main operators (selection and reproduction) need random procedures (Sivanandam & Deepa, 2008).
2. It has the ability to produce several solutions simultaneously (Kim and Zhang, 2003; Sivanandam and Deepa, 2008). Normally, the solution is the last generation and each individual of this generation represents a possible solution.
3. It provides acceptably good solutions for web searching problems within an acceptably quick time (Beasley et al, 1993a, p2.) This represents the main reason for using GA to solve the IR problem.
4. It can recombine different solutions to produce better ones via a crossover operator (Sivanandam and Deepa, 2008).
5. It is the best method used to solve a problem for which little is known (Vrajitoru, 1997; Sivanandam and Deepa, 2008).
6. It is a very general algorithm; therefore it works well in any search space (Vrajitoru, 1997).
7. It starts searching from multiple points instead of a single point, which avoids falling into local optima.
8. It works very well for a huge search space since processing is usually done on selected genes and not on all genes forming the search space.

Generally speaking, GA consists of three well known operators, namely, selection, crossover and mutation. And these are driven by the fitness function which favours specific

individual based on their fitness value. The first operator is the selection. This includes the selection of individuals forming the first generation which GA starts with as well as parent selection which is applied in later generations to prepare for crossover. The crossover is the process of producing offspring from the parents, while mutation is the last operator of GA in which one of the genes within the chromosome is replaced by other according to some mechanism.

The flow of the GA process is illustrated in Figure 3-8, while the overview of GA process showing the effect of fitness functions is presented in Figure 3-9. In Figure 3-8, c_{i1} represents chromosome one in generation i , g_1 is gene one of the chromosome, m is the length of the chromosome and n is the length of the generation, p_1 and p_2 are parent 1 and parent 2 which feed the crossover operator, O_i is offspring i which is the result of the crossover, whereas O'_i is the mutated offspring O_i . The behaviour of the GA mechanism is explained in the following subsections.

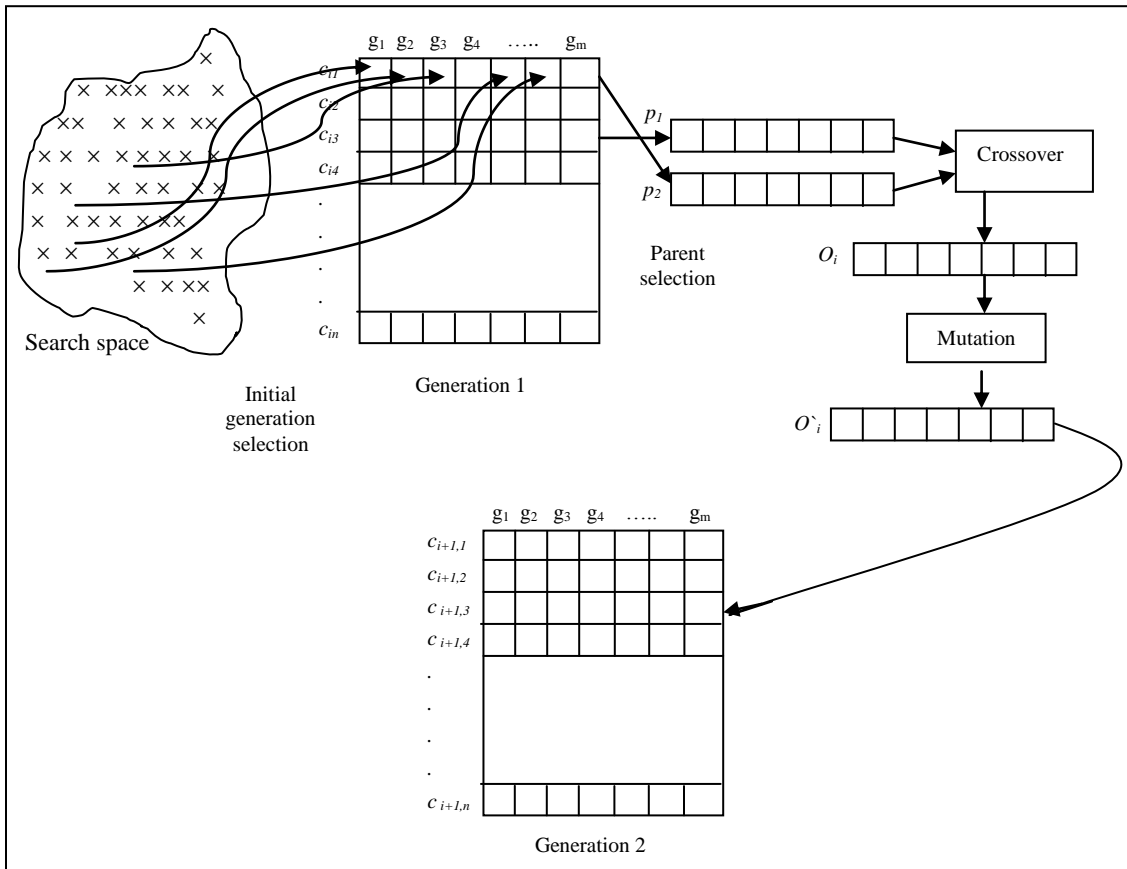


Figure 3-8: The flow of the GA process

3.4.1 Initial Generation

The population of GA in IRUGA consists of a set of documents. The first generation which is randomly created has a number of chromosomes. Each chromosome has a set of genes which are integers representing the document references.

The first generation of IRUGA has a vital importance since the genes of this generation form the base of the next generations, and most of its individuals have a high opportunity to be passed to the next generations. Therefore, special care must be taken when creating this generation. However, many authors (Pathak, Gordon and Fan, 2000; Aly, 2007; Yeh, Lin,

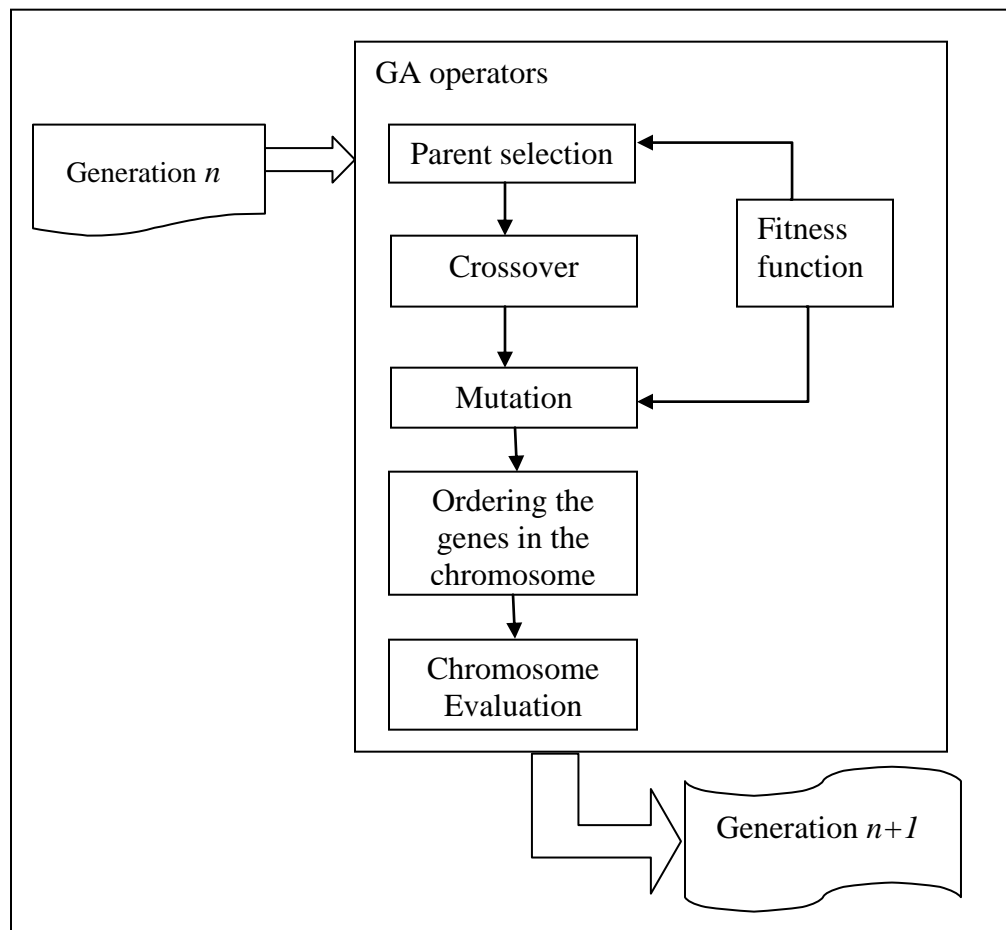


Figure 3-9: Overview of GA process showing the effect of fitness functions

Ke and Yang, 2007; Beasley et al, 1993a; Zhang, et al., 2005; Noreault et al, 1980; Horng and Yeh, 2000; Martín-Bautista and Vila, 1998; Lopez-Pujalte, Guerrero-Bote, and de

Moya-Anegon, 2003a) ignore this importance by selecting the individuals randomly without any favouring criteria. Although this method builds the population rapidly, the quality of individuals is low, causing slowness in finding the optimal solution by forming additional generations.

The proposed technique of creating initial generation of GA in IRUGA is the *selective random method*. In this method, genes are selected randomly but must satisfy two conditions in order to be added to the chromosome. The first condition is that the document added to the chromosome must include at least one keyword from the user query, i.e., document D is composed of set of terms t_i such that $D = t_1, t_2, \dots, t_n$, and a query $Q = q_1, q_2, \dots, q_n$, where q_i is the query keyword within the user query. Then the document D is selected such that $D \cap Q \neq \emptyset$.

The second condition for adding a document to the chromosome is that this document must not exist in the same chromosome, which means that genes within a chromosome are unique, i.e., given the chromosome $C = g_1, g_2, \dots, g_n$, and g_i is a gene within chromosome C , then $C \cap g_x = \emptyset$ where g_x is the newly added gene to the chromosome C .

Satisfying the first condition has advantages and one disadvantage. One of the advantages is that the quality of documents selected is high in the sense that it contains all terms queried by the user. In addition, including documents having this feature reduces the time for the system to converge or to find the optimal solution. On the other hand, the disadvantage of this selection criterion is that examining each document before adding it to the chromosome will slow down the process of creating each chromosome, but the time is compensated for by finding the optimal solution faster.

Adding documents having all query terms to the chromosomes seems at first glance to be the optimal solution because only documents that contain all query terms are selected, but in fact this is not enough to measure the relativity of the document. In fact, having all query terms within the document doesn't guarantee that it is relevant. Suppose that the query is "memory systems research". Although a document may contain all the terms, each word is surrounded by a different topic. Moreover, the GA will consider the term's frequency, in addition to the HTML tag that the terms fall in, beside other factors that will affect the document weight. Hence, selecting the documents that include all query terms is not enough

to determine the relevance but will reduce the search space and enhance GA performance by reducing the time for processing all search space and will produce a set of documents which is more likely to be related to the user query.

3.4.2 Parent Selection

Next generations are created by applying crossover between selected parents from the current generation. However, these parents must be selected in such a way that their good features are passed to the next generations.

Two techniques are used for selection. The first technique applied here is *elitism*, where the best individual is copied as it is passed to the next generation, guaranteeing that the best building block from this generation is inherited, e.g., given a population P_i , where i is the generation number, $P_i = c_{1i}, c_{2i}, \dots, c_{ni}$, where c_{ji} is chromosome j in population i , and $f(c_{ji})$ is the fitness value of the chromosome j in generation i , then $c_{xi} \in P_{i+1}$, such that

$$f(c_{xi}) = \max(f(c_{1i}), f(c_{ni}), \dots, f(c_{ni})).$$

The advantage of using elitism is that it can increase the performance of GA rapidly and meanwhile prevents losing the best individuals.

Other individuals of the next generation are created differently. From the literature, it is found that although Genitor selection and overlapping population selection show higher growth ratio than other techniques (Goldberg and Deb, 1991, p.70) (as shown in Section 2.2.5), this is not enough to be applied to GA unit of IRUGA as they are not able to beat the advantages of binary tournament selection. Hence, binary tournament selection is chosen in IRUGA for many reasons. The first one is that a wide range of individuals is selected due to its randomness. The second feature is that fit parents are selected because of tournament between the randomly selected individuals. The third advantage for using this technique is its low time complexity compared to the other types of selection (Goldberg and Deb, 1991, p.75). Finally, this method allows weak individuals to participate in the solution as will be illustrated later in this section.

In binary tournament selection, two individuals are selected randomly, and the one that has higher fitness value is picked up as the parent one. The same process is repeated to select the

second parent.

However, a slight change is made while implementing tournament selection in IRUGA, where two individuals are chosen at random from the population, and a random number r is then generated between 0 and 1. If $r < k$, (where k is a parameter set to 0.75 to favour the fitter individual), the fitter individual is selected (Al-Dallal and Abdulwahab, 2009). The idea behind this technique is to allow weak individuals to participate in the solution since they may include some good genes (documents) that have a degree of relevance and may enhance the results at some stages; therefore, this technique will provide a chance for these documents to be selected. Once the parents are selected, then they are passed to the next operator to perform crossover in order to produce the offspring of the next generation.

3.4.3 Hybrid Crossover Operator

Crossover is one of the GA operators used to produce a new generation. It is the process of producing offspring from two parents. Its purpose is to create new individuals having, hopefully, better performance than their parents. The most common form of crossover operator is applied with a predefined probability ranging from 0.75 to 1 to two selected individuals of a population to generate new offspring of the next generation. Indeed, several inherited features from the selected parents will be given to the resulting offspring (Vrajitoru, 1998; 2000).

Several crossover techniques have been discussed in Section 2.2.6 and it is shown that many of them have either one or more disadvantages.

In order to produce high quality offspring, some drawbacks have to be avoided in the proposed crossover technique while others could be overlooked or reduced by combining several techniques together. The main drawbacks that need to be avoided are generating lower performance offspring, breaking building blocks, generating offspring out of search space and low speed of convergence. These drawbacks are to be avoided in the proposed crossover which is called *hybrid crossover*.

3.4.3.1 The Design of Hybrid Crossover

The proposed crossover operator chosen to be implemented in IRUGA is a combination of

reordering crossover (Vrajitoru, 2000), fusion crossover (Vrajitoru, 1998) and one-point crossover (Marghny and Ali, 2005). When genes within a chromosome are ordered based on their fitness value and the order is important, then the crossover applied to such chromosomes is called a *reordering crossover*. In fact, the order of genes in the proposed crossover is important as it represents the ranked documents that will be displayed to the user. If one offspring is to be produced from the crossover process rather than two then it is called a *fusion crossover*. Combining these two techniques together and applying a one-point crossover on them forms the new crossover suggested in the GA unit of IRUGA.

In the one-point crossover, GA selects one point randomly to perform exchange of genes. A reordering crossover is applied to chromosomes having their genes ordered based on their fitness value from higher to lower. Since genes are in order within the chromosome then a 2-point crossover could not produce better results as the high quality genes are on the edges while exchange is done for the genes somewhere in the middle. Other techniques of crossover are not applied to the GA unit of IRUGA due to their disadvantages mentioned in Table 2-4.

The rationale behind using the ordered crossover technique over other techniques is the need to inherit the good genes and pass the good building blocks to the resulting offspring.

In fusion crossover (Vrajitoru, 1998) only one offspring is generated from the two selected parents. In this technique, the offspring inherits the genes from one of the parents with a probability according to its performance. The advantage of this technique is that the good genes of both parents are inherited simultaneously to the offspring, producing high quality offspring.

Combining the three techniques of crossover into one process allows fast convergence with high quality offspring. The ordered technique gathers the good genes into one side of the chromosome. Then the one-point crossover copies these gathered genes from the heavy side of both parents to one offspring only. This results in an offspring having the best genes of the parents.

3.4.3.2 The Functionality of Hybrid Crossover

The *hybrid crossover* operates in the following manner. Suppose there are two parents x and y of length L . These two individuals are selected randomly using binary tournament selection from current population P_i to produce one offspring O of population P_{i+1} . Firstly, the chromosome's genes are ordered based on their fitness value from higher to lower from the previous generation. Then a one-point crossover is applied by choosing crosspoint cp randomly over the range $[1.. L]$. The selected crosspoint divides the chromosomes into two parts. The first O 's genes $[O_0, \dots, O_{cp}]$ are copied from the candidate parent that has the greatest gene's value at position L_0 , which is x in the above mentioned example. The remaining genes of O are copied from the second parent starting from the leftmost position until the offspring O is filled up or until it reaches the specified location cp . Through the process of copying the remaining genes from the parents, the uniqueness of the copied gene must be considered, i.e., each gene can occur only once in the new offspring O . This is implemented by excluding the genes that already exist in O . When O is not filled up to the specified length, the fitness values of other genes in both parents are compared starting from $cp + 1$. The gene that has a higher fitness value contributes to O . This is done in order to generate offspring with appropriate genes from each parent and to guarantee that the length of O is maintained at L . Figure 3-9 gives an example of the proposed crossover in which numbers in each chromosome represent the fitness value of the gene at that position. The two candidates x and y that are shown in Figure 3-9-Step A are considered the contributors, and are selected from the previous generation using binary tournament selection explained in the previous section. The crosspoint cp is selected randomly to perform a one-point crossover. In this example it is 3. Because the first gene of x has a greater fitness value than the first gene of y , x 's genes along with the fitness values are considered as the first three genes of O . To complete the genes values of O , the other three genes are copied starting from the leftmost position of y . Then a competition between the genes in both x and y is done to complete the creation of O . Because the gene at position $cp+1$ in y has a greater value than that of x , then y 's genes are copied into O (step C in Figure 3-9). Once all positions in the offspring are populated with genes, these genes are ordered from higher to lower based on their fitness value (step D in Figure 3-9). The algorithm of hybrid crossover is illustrated in Algorithm 3-3.

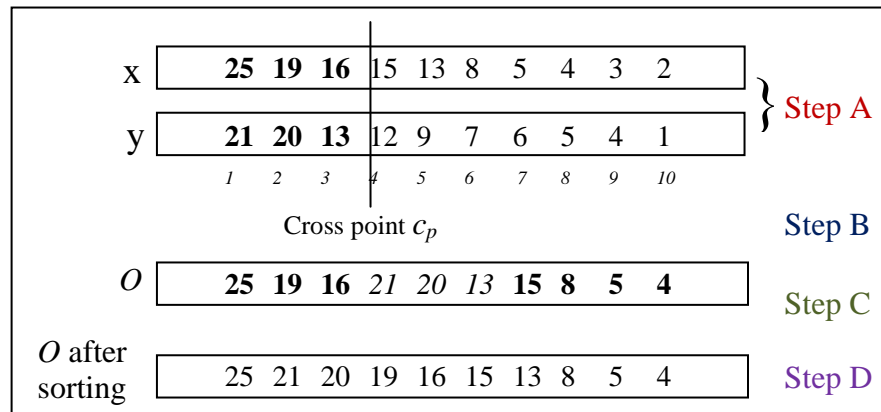


Figure 3-10: Illustration of the hybrid crossover process

Algorithm 3-3: The hybrid crossover operator

Prerequisite: Both parents are of same length and the genes in each of them are sorted with respect to their fitness value.

Select crosspoint cp randomly such that $0 < cp < \text{parent length}$.

$g_{\max} = \max \text{gene}(f(x_1), f(y_1))$ --compare fitness value of first gene in both parents

parent 1 = chromosome with g_{\max}

Create offspring such that: O

= $g_1, g_1 \leq cp$

= $g_2, g_2 \leq cp, g_2 \notin O$ and $\text{length}(O) \leq \text{length}(\text{parent1})$

If $\text{length}(O) < \text{length}(\text{parent1})$

begin

$g_{\max} = \max \text{gene}(f(x_{cp+1}), f(y_{cp+1}))$

parent 1' = chromosome with g_{\max}

Copy genes from parent 1' to O such that genes are unique in O

end;

Order genes in O in descending order with respect to their fitness value.

3.4.4 Mutation

Mutation is the last genetic operator used in the GA unit of IRUGA. In mutation, one or more genes are selected randomly to be replaced by other genes according to some criteria. It causes the individual genetic representation to be changed according to some probability p_m ranging from 0.001 to 0.7. Because of its importance and effect on the generated chromosome, it is applied in this system with probability of 0.7.

Mutation is generally considered to be a background operator since it ensures that the probability of searching a particular subspace of the problem space is never zero (Minaei-Bidgoli and Punch, 2003). Although it is applied with low probability, it is considered as a very important operator (Beasley et al, 1993b). Applying mutation during GA achieves

many objectives. These objectives are: to restore lost data, to explore variety of data (Radwan et al 2006), to improve diversity of the solution (Noreault et al, 1980) and finally but not last to reduce the possibility of converging to a local optimum (Minaei-Bidgoli and Punch, 2003; Noreault et al, 1980).

The mutation technique applied in the GA of IRUGA is the most common technique of mutation (Radwan et al 2006; Aly, 2007; Alfonseca, 1991; Vrajitoru, 1998; 2000; Beasley et al, 1993a; Martín-Bautista and Vila, 1998; Klabbankoh and Pinngern, 2008; Lopez-Pujalte, Guerrero-Bote, and de Moya-Anegon, 2003a) and is implemented by randomly selecting one gene and changing it with another value from search space such that the new gene has better or the same fitness value and does not exist in this chromosome. This is to ensure that at least the same performance as the replaced gene. This technique of mutation is more suitable for the developed system when compared with other methods of mutation in literature (listed in Table 2-4). Examples of inapplicable techniques of mutation are: selecting 2 random genes and exchanging their locations (Asllaniand and Lari, 2007). This method is useless since genes are reordered based on fitness value, and hence will return to the same original position and this will not change anything in the chromosome. Arithmetic mutation (Beasley et al, 1993b) and introducing Gaussian noise (Pathak, Gordon and Fan, 2000; Song and Park, 2009; Steinbach, Karypis, and Kumar, 2000) are discarded from this work as they require extra calculation load and will slow down the process of generating the chromosomes. In addition, they may produce genes that are out of search space. Similarly, using inversion of a weight of mutated gene (Horng and Yeh, 2000) will reduce the system performance due to huge amount of calculation and it requires previous knowledge about all genes prior to implementing it.

An example of the mutation applied in this work is illustrated in Figure 3-10 where the numbers in this figure represent the fitness value of genes at these positions. The chromosome represented here is a continuation to the one shown in Figure 3-9. The position of mutation is selected randomly (position 7 in this example – Step B). The gene at this

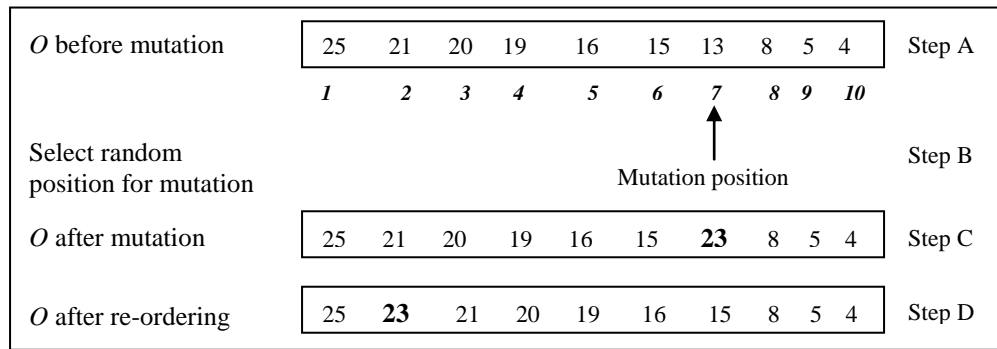


Figure 3-11: Illustration of the applied mutation in IRUGA

position is replaced by another gene selected randomly from the space such that it has a better fitness value or the same as the replaced one. In this example, the new value is 23 and it is better than the original one: 13 (Step C). This new value is unique within this chromosome; therefore it is exchanged with the original one. Then genes of this chromosome are re-ordered in descending order according to their fitness value to produce the new chromosome (Step D).

3.4.5 Fitness Function

Fitness function is a performance measure or reward function that measures the relevance of the documents to the user query. The decision about whether to accept or reject a document for crossover or mutation depends only on the value computed by the fitness function. This function is used in the GA process to evaluate the documents while selecting parents to perform crossover and mutation. The evolution process results in pushing high quality individuals to survive over lower ones.

From the literature review, it is deduced that the fitness functions can be categorized into three types, namely, the *terms weight-based* fitness function, *similarity-measuring* fitness function, and the *custom* fitness function.

Furthermore, the fitness functions consist of a set of factors. These factors can be classified into *statistical* factors, *formatting* factors, and *semantic* factors. From another point of view, these factors can be classified into local factors and global factors. Details of these categories and types are explained below.

3.4.5.1 *Fitness Function Categories*

The fitness functions developed by researchers can be categorized into three categories:

The first category uses the *term weight* as an evaluation function to the document. In this category the document is evaluated by taking the summation of the query *term weight* (Kim and Zhang, 2003; Billhardt et al, 2002; Cummins and O’Riordan, 2006; Vrajitoru, 2000; Radwan et al 2006; Aly, 2007)

The second category is the *similarity* function which measures the distance between the document and the query vector. However, this method is most suitable for documents indexed using the vector space model, and doesn't fit into the proposed model because it uses the enhanced inverted index model (Klabbankoh and Pinngern, 2008).

The third category is the *custom* fitness function in which the fitness functions are developed using set of factors that best suit each model (Marghny and Ali, 2005; Picarougne et al, 2002a; Fan et al, 2004).

3.4.5.2 *Types of Factors Used in Fitness Functions*

Normally, the fitness function consists of a set of factors. These factors can be divided into three types: statistical, formatting, and semantic factors.

Statistical factors are those obtained from the document set by counting certain elements, such as frequency of query terms within a document or within the document set, frequency of unique terms within a document or within the document set, total number of documents in the set, or frequency of the total number of terms in a document or in the set.

If the fitness function is built using statistical factors only, then the documents become relevant with respect to other documents in the set. Therefore, these functions are applicable only to the document set under consideration and cannot be generalized. Moreover, the best retrieved and ranked documents need not be purely relevant because they have the best score among others in the set, and at the same time could be of low relevance.

Formatting factors are the second type of factors which include bold, italic, underlined, and text emphasized by using different font and size. However, identifying such format requires special tags to handle them and a special indexing model that is able to detect such

format and treat terms accordingly.

Semantic factors are the third type of factors. These factors can be obtained using semi-structures, e.g.: HTML and structured documents, XML and Resources Description Framework (RDF) (Karthik et al,2008). In such documents, special tags are used to fragment the document into different meaning units such as title, header, anchor (link to other document), table, and normal text. Since most researches mentioned in the literature review chapter use non-structured documents, these types of factors are not included in their fitness function. By assigning different weights and counting the frequency of such terms, these factors become a combination of statistical and semantic factors.

From another perspective, the factors used to evaluate the documents can be classified into two categories: *local factors* and *global factors*. These categories are explained in the following sub-section.

3.4.5.3 Local Factors Verses Global Factors

The fitness function can have both local and global factors or be restricted to one of them only. *Local factors* are those obtained from the document under consideration such as document size, number of unique terms within the document and frequency of a specific term within the document. On the other hand, *global factors* are those obtained from the document set such as total number of documents in the set, total indexed terms, total number of a specific term within the document set and total number of documents indexing a term (Minaei-Bidgoli and Punch, 2003).

Local factors are preferred for many reasons. The first one is that the document is evaluated independently of other documents in the set. The second one is that obtaining the factors requires less time since these are obtained for the document in consideration only, while obtaining global factors requires processing all documents in the set which adds extra time and processing load on the system. The third is that by setting a threshold for the fitness value, the relativity of a document to the query is measured more accurately by stating how relevance the document is to the query.

However, the time consumed and additional processing load of obtaining some global factors can be reduced if they are evaluated while constructing the index and stored before

accepting the user query so that it can be referenced whenever needed without affecting response time to user query.

In addition to the extra time and processing load required to obtain the global factors, the retrieved documents are only relatively relevant in comparison to other documents in the set and may not be purely relevant. Hence, fitness functions using only global factors cannot be generalized.

Based on the above categorization of the fitness functions and the types of factors, two fitness functions which fall under the custom category according to the first classification are developed. The term's weight is considered as a component of the fitness functions, and similarity functions are avoided in this work as our model does not use the vector space model in indexing the terms. The first fitness function uses a combination of local and global factors and the second one uses only local factors. However, the first one uses only statistical and formatting factors, while the second uses statistical, formatting, and semantic factors.

3.4.5.4 Multi Terminal Fitness Function

The first fitness function developed and to be tested in IRUGA is called the *multi-terminal* fitness function. This function consists of many factors: some are local and others global. The user query is considered as a global terminal (Cummins and O'Riordan, 2006). Also it consists of statistical and formatting factors.

Starting from the common TF-IDF formula where the term weight is evaluated as:

$$w_i = f_{ij} \times \log\left(\frac{N}{df_i}\right)$$

To include the percentage of the term frequency within the document, first components will be divided by document length F_i to be $\frac{f_{ij}}{F_j}$ which forms the first component of the proposed formula. In addition, there is a need to include the distribution of the query keywords among

the search space, so the component $\log\left(\frac{T}{r_i}\right)$ is included, which evaluates the log of total number of terms in the collection by the total number of the term under consideration. To include the HTML tag weight h_{ij} , it is multiplied by the above components.

This fitness function that evaluates the document consists of three components. The first one is the ratio of existence of a keyword in the document to the total unique query keywords, or in other words, how many keywords of the user query exist in the document. The second component is the ratio of query keywords that exist in the document to the unique number of terms in the document. The third component is the summation of the weight of the query keywords that exist in this document. Thus, the fitness function is presented in formulas 3.1 and 3.2 whereas the explanation of its terms is presented in Table 3-2:

$$f(d_j) = \frac{k_i}{K} \times \frac{k_i}{t_j} \times \sum_{i=1}^K w_i \quad (3.1)$$

$$w_i = \frac{f_{ij}}{F_j} \times h_{ij} \times \log\left(\frac{N}{df_i}\right) \times \log\left(\frac{T}{r_i}\right) \quad (3.2)$$

Bearing in mind that the fitness value for each chromosome c_i in the population is the summation of the fitness value for each document in the chromosome and is calculated using the following fitness function:

$$F(c_i) = \sum_{j=0}^{length} f(d_j) \quad (3.3)$$

where *length* represents the maximum length of chromosome c_i .

The description of each factor of the fitness function is presented in the following, starting by the term weight function which describes the formula in (3.2).

The first component of the term's weight is: $\frac{f_{ij}}{F_j}$. This factor is the division of the frequency of the term i in document j by the size of document j . So it represents the probability of the term i in document j . By using this factor, the fitness value is not biased by document size

and is not dependent on it. If this factor is not used, then considering the term's frequency

Table 3-2: Terms of formulas 3.1 and 3.2 showing their description, domain and type.

Terminal	Description	Domain	Type
i	Term i in the document.	Local	-
j	Document j in space.	Local	-
w_i	Weight of term i in document j	Local	Format – Statistical
k_i	Frequency of term i in user query Q .	Global	Statistical
K	Total number of terms in Q .	Global	Statistical
f_{ij}	Frequency of term i in document- j .	Local	Statistical
F_j	Size of document j (total number of words in document- j)	Local	Statistical
t_j	Number of unique terms in document j	Local	Statistical
N	Total number of documents in space.	Global	Statistical
df_i	Total number of documents having term i .	Global	Statistical
T	Total number of all terms in space.	Global	Statistical
T_i	Total number of term i in space.	Global	Statistical

regardless of document size could be biased by the high frequency in big documents so that more occurrences of the term in such documents will give the impression that this document is more relevant to user query while this high frequency could represent a small fraction of that document and doesn't represent the actual relation to this keyword. Therefore considering the probability of the term within the document is important, because it reflects the level of relevance of the document to the keyword.

The second component in term's weight is the HTML tag weight w_i . It is a weight given to a term depending on the HTML tag that this term falls in. Details of the HTML tag weights are described in Section 3.2.2. The weights associated to each HTML tag are listed in Table 3-1.

The third component in the term's weight formula is $\log\left(\frac{N}{df_i}\right)$. This component represents the IDF component of the common Salton and Buckley weighting scheme (Salton and Buckley, 1988). It is the logarithm of the total number of documents in the space divided by the total number of documents indexed by term i (Radwan et al 2006; Noreault et al, 1980;

Cummins and O'Riordan, 2006).

The fourth component of the term weight formula is $\log\left(\frac{T}{T_i}\right)$. It is the logarithm of the total number of terms in the space divided by the total occurrences of term i in the search space (Noreault et al, 1980).

The above four components are used to obtain the term's weight. In order to obtain the document's weight, these components are multiplied by two more components. The first one is $\frac{k_i}{K}$. This component is used to obtain the ratio of the query keywords exist in the document corresponding to the total query keywords. Thus, this component will be one for the documents that have all user query terms, and less than one otherwise.

The second component used to obtain the document's fitness value is $\frac{k_i}{t_j}$. This component is used to obtain the probability of the term among all unique terms in the document under consideration. So this component depends on the unique terms within the document.

Thus, this function returns high fitness value if the document is relatively small and has a small number of unique terms.

3.4.5.5 Term Proximity Fitness Function

The second fitness function to be tested in this work is called: *Term Proximity Fitness Function (TPFF)*. This function shows much better performance than the multi-term fitness function explained in the previous section and will be used though out this thesis. That is because it has many advantages. These features are:

1. It utilizes the *term distance*.
2. It includes only local factors.
3. It uses all the three types of factors: statistical, formatting and semantic.
4. It has a maximum upper limit; hence a threshold can easily be set to determine the relevant documents.

The TPDF function is defined in formula 3.4 shown below and its terms are explained in Table 3-3:

$$f(D) = a \frac{\sum_{i=1}^K k_{ui}}{K} + b \frac{\sum_{i=1}^K k_{ui} - 1}{\sum_{i=1}^{K-1} \min(d_{i,i+1})} + c \frac{1}{\text{avg}(\sum_{i=1}^K \min(p_i))} + d \log\left(\frac{\sum_{i=1}^K w_i}{\sum_{i=1}^K k_i}\right) \quad (3.4)$$

Table 3-3: Terms of formulas 3.4 showing their description, domain and type

Terminology	Description	Domain	Type
k_{ui}	Represents the existence of k_{ui} within the documents	Local	Statistical
K	The query length, i.e. the total number of terms in the query	Local	Format – Statistical
$d_{i,i+1}$	Distance between term i and term $i+1$ of the query terms	Local	Semantic
p_i	The offset (position) of term i within the document	Local	Constant
F	Document size (total number of terms in the document)	Local	Statistical
w_i	Weight of term i in the document as per Table 3-1	Local	Format – Statistical - Semantic
a, b, c and d	Weighting factors for each component	Local	Constant

This function is a summation of four components: the first one is the ratio of the existence of the query's keywords within the document, where k_{ui} represents the unique existence of keyword i within the document D . In other words, this component reflects how many of the query keywords exist in the document divided by the query size. This factor has a maximum value of one. Further explanation for computing this factor; assume “web data mining” is the requested query that is entered by the user. If D has just two keywords such as “web” and “mining” and $K=3$ (i.e. query size), then this factor will be equal $2/3$. This factor equals $3/3$ when D has all the assumed keywords (i.e. “web”, “data” and “mining”).

The Minimum Term Distance (MTD) between query keywords within document D is used to compute the second component of the evaluation function. However, this component is evaluated by subtracting one from the total number of existence of query's keywords within the document D . Consequently, the resulting value will be divided by the $\sum_{i=1}^{K-1} \min(d_{i,i+1})$. Undoubtedly, $\sum_{i=1}^{K-1} \min(d_{i,i+1})$ represents the summation of the minimum (shortest) distance between query keywords in the document D . The reason for subtracting one here is that the distance between K keywords is $K-1$. Recall to the above

example for the suggested query (i.e., $q=\{\text{“web”}, \text{“data”}, \text{“mining”}\}$), MTD equals 2. Indeed, this component will return 1 if all query keywords exist in the document and they appeared adjacent.

The third component depends on the position of MTD within the document. It represents the reciprocal of the average of the minimum distance between query terms

$$avg(\sum_{i=1}^K \min(p_i)).$$

The highest value of this component is given when the keywords appear right at the beginning of the document, such as in the title, header or in the first sentence of the document. However, the maximum value of this component is one only if the query consists of one word and this word is the first word in the document. Otherwise, the value is always less than one as it considers the average offset of the first appearance of MTD keyword. The value of this component decreases as the keyword appear far from beginning of the document. The second and third components are further clarified by the following example:

Example 1: to understand the way of calculating component 1, 2 and 3, assume a document d of length 12 that has the following string of words:

ABCDEFGHIJJD

and assume the query is CDF, while the offset (position) of these words within the document are 3, 4, 6, 9 and 12. The first component of the Formula 3.4 is

$$\frac{\sum_{i=1}^K k_{ui}}{K} = \frac{3}{3},$$

where the 3 in the numerator represents the number for unique query keywords

that exist in the document d , and here all the three keywords are exist in the document d , while the 3 in the denominator represents the query size. The MTD of the second component is calculated by $\sum_{i=1}^{K-1} \min(d_{i,i+1}) = \text{MTD} = \min(C, D) + \min(D, F) = 1+2 = 3$.

The factor k_{ui} of this example is equal to 3, where all query keywords appear in the document. Then, the results of the second component will be summarized as follows

$$\frac{\sum_{i=1}^K k_{ui} - 1}{\sum_{i=1}^{K-1} \min(d_{i,i+1})} = \frac{3-1}{3} = \frac{2}{3} = 0.667 \quad \rightarrow \text{Final result of the second component}$$

In order to calculate the third component, which is the average position of minimum distance between query terms, it needs to get the position of the terms used in obtaining the above MTD. In this example they appear in the 3rd, 4th and 6th position. Therefore:

$$\text{avg} \left(\sum_{i=1}^K \min(p_i) \right) = \frac{(3 + 4 + 6)}{3} = 4.33$$

$$\frac{1}{\text{avg}(\sum_{i=1}^K \min(p_i))} = \frac{1}{4.33} = 0.231 \quad \rightarrow \text{Final result of the third component.}$$

The last component considers the HTML weight of the keywords. It is evaluated by finding the log of the average term's weight of the query keywords in the document. This part reflects the importance of the query keywords within the document. If the keyword is very important, such that it appears in either title, subtitle, or header tag, or it appears in a text that is emphasized using bold or italic tag, or it is within an anchor text (text that refers to other web document) then this component will have a high value according to the tag weights specified in table 3-1. Indeed, the maximum value of this part is 1 if the total weight of the keywords is 10 times greater than the frequency of these keywords. In order to normalize this component to cope with the other components of this function where each one has a maximum value of one, the *log* function is applied to the average keyword weight in order to reduce this value of this component to be smaller than one. This will control the upper limit of it.

The four components of this function are not of equal weight, since each one reflects a degree of relativity to the user query. Therefore, a high importance is given to the components that reflect high relativity of the document to the query and able to distinguish one document from the other. To achieve this, each component of this function is multiplied by a weighting coefficient according to its importance, such that the summation of these weighting coefficients is one. In this function, the first component is given a high importance in reflecting the relativity of the document, where the document is supposed to be more relevant if it refers to the all query keywords; therefore, a weight of 0.3 is given to coefficient *a*. Moreover, if these keywords are adjacent within the document, this yields

high degree of relativity; hence, a weight of 0.3 is given to coefficient b as well. The third component is given a high importance because it favours one document over the other when the first two components are achieving the same score. This factor gives high importance to the document that refers to the query keywords that appear right at the beginning of the document, such as the title or first paragraph. Therefore, this component is given high weight as well. Thus, a weight of 0.3 is assigned to coefficient c . On the other hand, the fourth component can produce same results for multiple documents depending on the frequency of the query keywords and the HTML tag that these words appear in which may not distinguish a document accurately. Even though, this component is included in the function in order to benefit from the HTML tag weight and the frequency of the keywords. Therefore, this component is given a lower weight and d is set to 0.1.

3.5 Success Criteria of Relevant Document

In similar researches, the proposed techniques are tested against readymade data sets where the queries and relevant documents are predefined. Examples of such sets are: CRAN (Billhardt et al, 2002; Salton and Buckley, 1990), CISI (Radwan et al 2006; Vrajitoru, 1997; Billhardt et al, 2002; Cummins and O’Riordan, 2006; Aly, 2007; Salton and Buckley, 1990), CACM (Radwan et al 2006; Billhardt et al, 2002; Aly, 2007; Vrajitoru, 1997; Salton and Buckley, 1990), OHSU90-91 (Cummins and O’Riordan, 2006), and TREC (Kim and Zhang, 2000; 2003; Yeh et al, 2007; Kim and Croft, 2008, Xu et al, 2008, Uematsu et al, 2008). Hence, these researches relay on the relevance of document as stated by the provided document set and doesn’t mention a success criterion. However, in this work the data set applied is not provided with queries and their relevant documents. Therefore, a set of queries and their corresponding relevant documents are created manually for this thesis. While creating this set, a document is judged as relevant when satisfying two conditions, which are:

1. The document must contain all the query words. Given a document D , and a query

$$Q = \{q_1, q_2, \dots, q_n\}, \text{ then, } D \cap Q = Q.$$

2. The query words must appear at least once in adjacent place with the document. Given a query Q of length k , then MTD for $Q = k-1$.

According to the analysis of retrieved documents based on the Term Proximity Function, it is found that the document is relevant and can be included in the solution if it has a minimum fitness value of 0.6. This is explained as follows. If the document contains all query keywords, then the first component of TPF will equal one. Similarly, when the MTD = $k-1$, then the value of the second component will equal one as well. Since each one of these components has a weight of 0.3 and these two components are summed in the formula, this leads to minimum fitness value of 0.6. Therefore, the threshold fitness value for accepting a document as a relevant to the query is 0.6.

As an application of the TPF on the data set, consider the query: “Digital systems design”. This query has 44 relevant documents as per the success criteria defined aforementioned while 170 documents in the collection reference all keywords. Figure 3-12 shows the document that is retrieved in the first position as the most relevant document. As this document referenced the three keywords, the first component of TPF will be:

$$\frac{\sum_{i=1}^K k_{wi}}{K} = \frac{3}{3} = 1$$

The second component utilizes the MTD of the keywords. MTD for these keywords in this document is 2 as these keywords appear adjacent at least once, hence

$$\frac{\sum_{i=1}^K k_{wi}^{-1}}{\sum_{i=1}^{K-1} \min(d_{i,i+1})} = \frac{3-1}{2} = 1$$

By looking at the position where first occurrence for the MTD appear, it is shown that it appears in the header1 <h1> that is few lines after the beginning of the document. By excluding the words out of the HTML tag (as per the indexing algorithm 3-2) and the HTML tags, it is found that these keywords appear at locations 4, 5 and 6. Hence the this component of TPF will be

$$\frac{1}{avg(\sum_{i=1}^K \min(p_i))} = \frac{1}{\frac{4+5+6}{3}} = \frac{1}{5} = 0.2$$

The last component of TPF considers the HTML tag weight. By refereeing to table 3-1, the weight of the keywords is the summation of weight of each keyword and will be:

Weight of “digital”= $w(h1) = 5$,

Weight of “systems”= $w(h1) = 5$,

Weight of “design”= $w(h1) = 5$, hence $\log\left(\frac{\sum_{i=1}^K w_i}{\sum_{i=1}^K k_i}\right) = \log\left(\frac{5+5+5}{3}\right) = 0.699$

Multiplying each component by the weighting factors and adding them up yields:

$$0.3(1) + 0.3(1) + 0.3(0.2) + 0.1(0.699) = 0.7299$$

Another example can be considered here to show the output of the TPF when the document is less relevant. The document shown in Figure 3-13 will be considered. This document references the three keywords; however, these keywords are not adjacent.

By following same analysis for each keyword within this document, it is found that first component has value of 0.3 (when multiplied by the weighting factor), second component also have value of 0.3, sine the three keywords are adjacent; however, the first occurrence of the MTD is at offset 32, 33 and 34, giving average of 33, so third component will be the reciprocal of 33= 0.0303, when multiplied by the weighting factor it becomes 0.0909. Finally is the HTML weight component. The total HTML weight of these keywords is 32, yields $\log\left(\frac{33}{12}\right) = 0.426$. Multiplying this value by the weighting factor it becomes 0.0426.

so the fitness value of this document is

$$0.3 + 0.3 + 0.0909 + 0.0426 = 0.6517.$$

```
Date: Mon, 02 Dec 1996 15:02:00 GMT
Server: NCSA/1.4.2
Content-type: text/html

<HTML>
<head>
<title>CSE477 Announcements</title>
</head>

<h1>CSE477: Digital Systems Design</h1>
<h3>Steve Burns, Spring 1996</h3>

<hr>

<b>April 3</b>
<ul>
<li>If you are interested in obtaining your own evaluation module for
the 68HC11, you can purchase one directly from Motorola with a heavy
student
discount. Send a copy of your student ID and a check from $68.11 to
<pre>
Motorola University Support
Suite 100
505 Barton Springs Road
Austin, TX 78704
</pre>
to receive a S68HC11EVBU. Allow 2 to 3 weeks for delivery. (Note that
this board is *not* the same as the one we use in the lab.)
</ul>
</body>
<address>
<hr>
burns@cs.washington.edu
</address>
<p>
</html>
```

Figure 3-12: first document retrieved by query: Digital systems design

Although the frequency of the keywords in this document (12 occurrences) is greater than the previous document (3 occurrences), it has lower fitness value than the one in previous example because the first occurrence of MTD is at position 33 compared to 5 in previous example. Also the keywords are appearing in lower HTML tags which are “a” and “body” that have weights of 4 and 1 respectively, compared to “h2” which has weight of 5.

```

.<title>Gupta, Rajesh</title>
<!-- Changed by: Scott R Stonefield, 15-May-1995 -->
<h1>
Rajesh K. Gupta

</h1>
<h2>Areas of research interest:</h2>
<ul>
<li>
<h4>
<li>Computer-aided Design of Digital Systems</a>:
High-Level Synthesis and Optimization Techniques; Synthesis for Embedded Systems
; Combinatorics
<li>Computer Architecture & Systems</a>:
Multiprocessor Analysis and Design; Heterogeneous Systems;
Microprocessor Organization and Tradeoffs for Integrated Implementations
<li>Real-time Computing Systems</a>:

Constraint Modeling; Analysis Scheduling VLSI and Systems Design;
Circuit Structures for Computer and Communication Systems;
Integration of Computer
and Communication Systems; Designs for Low Power Applications
Communication and Signal Processing: DSP Algorithms and Implementations; Broadband Communication

</li>
</ul>
<h2>Research Group:</h2>

<h2>Administrative help:</h2>
<ul>
<li>
<h4>
<li>Ronnie Howard</a>
</li>
</ul>

<h4>More Info:</h4>
<li>Personal Home page.</li>

<dt>2214 Digital Computer Laboratory
<dt>1304 West Springfield Avenue
<dt>Urbana, IL 61801
<p>


<dt>(217) 244-6025
<dt>(217) 333-3501<i>- fax</i>
<dt>rgupta@cs.uiuc.edu
</h4>

<h4>

Go back to Faculty Index</a></h4>

```

Figure 3-13: example of low relevant document for the query: Digital system design"

Rajesh K. Gupta 

Areas of research interest:

- [Computer-aided Design of Digital Systems](#): High-level Synthesis and Optimization Techniques; Synthesis for Embedded Systems ; Combinatorics
- [Computer Architecture & Systems](#): Multiprocessor Analysis and Design; Heterogeneous Systems; Microprocessor Organization and Tradeoffs for Integrated Implementations
- [Real-time Computing Systems](#): Constraint Modeling; Analysis Scheduling VLSI and Systems Design; Circuit Structures for Computer and Communication s Systems; Integration of Computer and Communication Systems; Design for Low Power Applications Communication and Signal Processing; DSP Algorithms and Implementations; Broadband Communication


Research Group:


Administrative help:

- [Broonie Howard](#)

More info:

[Personal Home page](#)
2214 Digital Computer Laboratory
1304 West Springfield Avenue
Urbana, IL 61801
(217) 244-6025
(217) 333-3501 - fax
rgupta@ca.sruc.edu



 [Go back to Faculty Index](#)

3.6 Environmental Settings

The design of the two units of IRUGA (Information Retrieval Using Genetic Algorithm) model is explained in the previous sections. When examining IRUGA, several aspects need to be addressed. First of all, the environment configuration of the machine used to run IRUGA as well as the programming tool used in implementing this model is described.

This chapter is organized as follows: the environment configuration and programming language are described in Section 4.2. Section 4.3 examines experimentally the parameters of the GA unit of IRUGA. Section 4.4 explains the method to be followed in comparing the proposed techniques with the existing techniques on implementing each operator of the GA unit of IRUGA. Finally, this chapter is summarized in Section 4.5.

3.6.1 Hardware Configuration

IRUGA is implemented using IBM laptop powered by Intel Core 2 Duo CPU @ 2.35GHz having 3GB RAM. The database is stored on a SUNW SPARC-Enterprise machine.

3.6.2 Software Configuration

The software configuration concentrates mainly on the programming languages used to implement IRUGA.

In fact, there was a discussion about the programming language suitable for implementing IRUGA. After encountering several limitations in C++, Oracle database 10g was chosen to implement IRUGA using PL/SQL programming language. It was adopted for this approach due to the following features:

1. The integration of PL/SQL with SQL engine is much faster than that on C++ (Application development: PL/SQL, Java or C++?, 2002).
2. The ability to manage thousands of documents without having to worry about space because of the function of internal tables and indexes. This is one of the limitations faced due to the data structure adopted in implementing this model, which obliged us to think of a better alternative
3. Data access is very fast and not sequential as in C++, since the data structure used in C++ is the *link list* and requires sequential processing of all nodes in this list until reaching the desired node.
4. The built-in functions save many lines of codes, such as *order* function and *comparison* function.

3.7 Parameter Settings of the GA unit of IRUGA

Next, is to perform a set of experiments to evaluate the performance of IRUGA. These experiments are of two types. The first type is a set of experiments conducted to find the best setup for IRUGAs' parameters. These parameters include population size (ps), chromosome length (cl), crossover probability (c_p), mutation probability (m_p), and the termination criteria. The second type of experiments is those applied on the GA unit of IRUGA to test its actual performance where the output is evaluated using the recall and precision measures. The experimental work for choosing IRUGA's parameters is presented in this section while the experiments used to evaluate IRUGA are explained in the following section, while the experimental results are analyzed in the next chapter. The evaluation of the experimental work is conducted in terms of the following criteria:

1. To see how IRUGA scales-up with population size (ps).

2. To find the best setup for IRUGAs' parameters, i.e. ps , chromosome length (cl), crossover probability (c_p), mutation probability (m_p), and the termination criteria.
3. To see how the performance of IRUGA is influenced by these parameters.

In order to have an efficient setup for the GA unit's parameters, a success rate is computed in this experiment. The success rate is evaluated by testing the GA unit with 10 independent runs for 4 queries of different lengths, i.e. $2 \leq l \leq 5$, where l is the query length. These runs are applied on different values of ps and cl . The quality criteria for a sufficient solution are obtained by ignoring the solutions that are far from the optimal. The accepted solutions are those which have fitness values varying between 0.9 and 1.

3.7.1 Choosing the Appropriate Population Size

Since GA is a probabilistic algorithm, there is no sharp edge for specifying the population size of GA. Moreover, the population size may be influenced by the size of possible solution for a given problem. Hence an empirical study is conducted in order to specify the suitable population size of the GA unit in IRUGA. This is done by examining the GA unit with variant values of population sizes (ps), i.e.: $ps \in \{75, 100, 125, 150\}$. The evaluation is achieved by calculating the probability of success per generation for each population size.

Figures 3-11 to 3-14 present the performance curves for different values of ps , i.e. $ps = 25, 50, 75$ and 100 . Each curve is based on 10 independent runs. The curves of these figures show the probability of success of solving the problem by generation i . In this experiment, in order to find the precise value of ps , 10 queries of different length $2 \leq ql \leq 5$ are examined. Summary of these results are presented in table 3-4.

3.7.2 Chromosome Length

Each chromosome represents a possible solution which is a set of possible documents relevant to the user query. Each chromosome consists of a set of genes that form a possible

solution to the user query. Therefore, its length must be selected such that it is able to

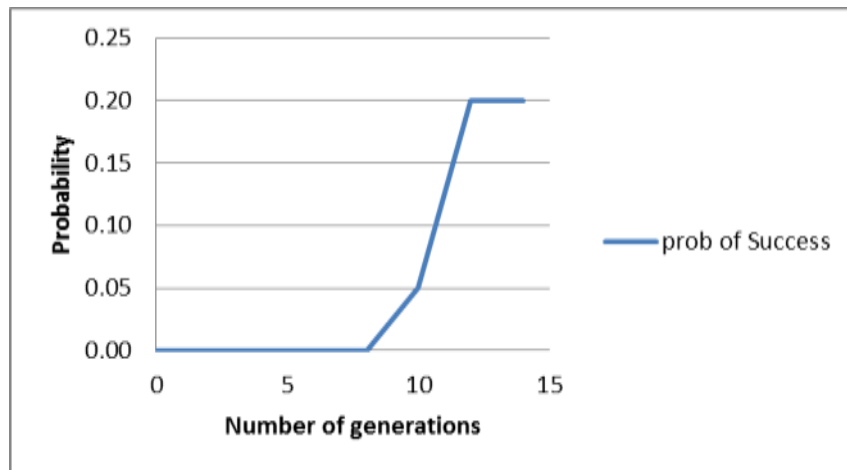


Figure 3-14: Probability of success for population size = 50 and chromosome length = 50.

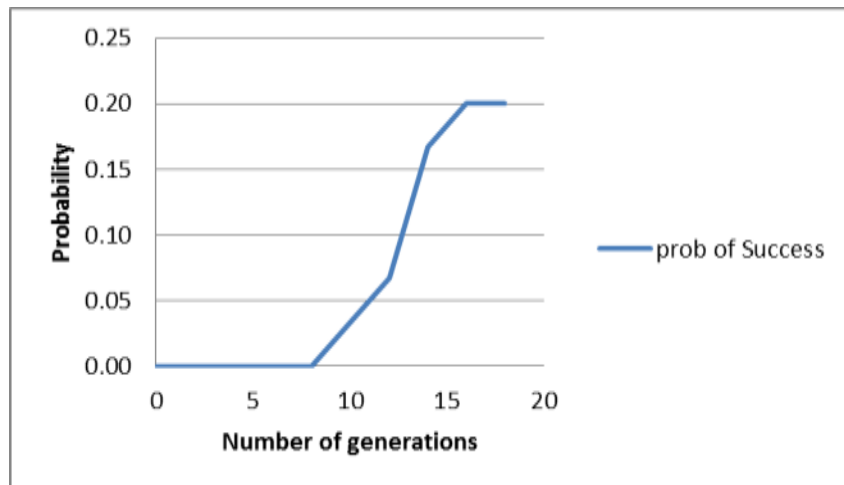


Figure 3-15: Probability of success for population size = 75 and chromosome length = 50.

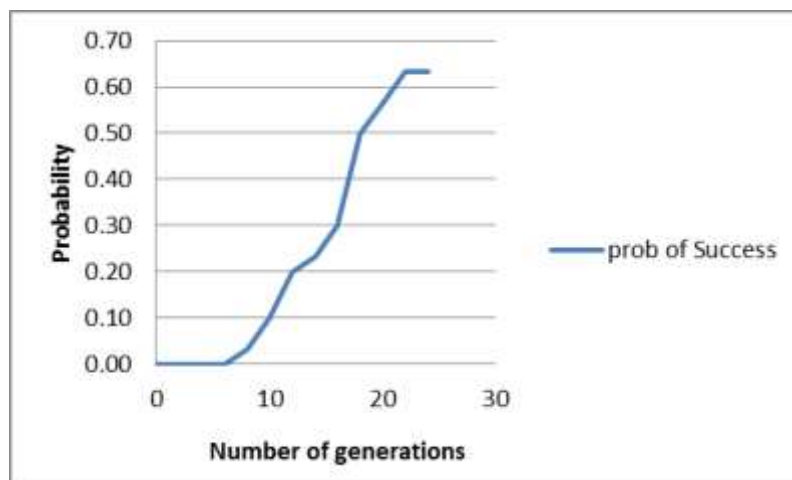


Figure 3-16: Probability of success for population size = 100 and chromosome length = 50.

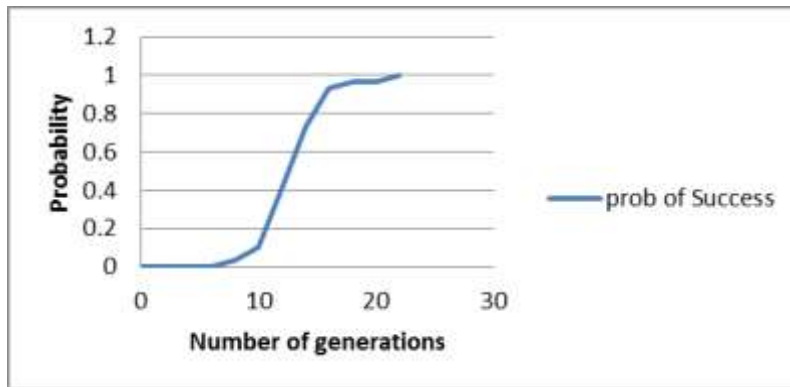


Figure 3-17: Probability of success for population size = 125 and chromosome length = 50.

Table 3-4: Probability of success for different values of the population size

Generation Number	ps=50	ps=75	ps=100	ps=125
2	0	0	0	0
4	0	0	0	0
6	0	0	0	0
8	0	0	0.03	0.03
10	0.1	0.1	0.10	0.10
12	0.2	0.2	0.20	0.40
14	0.2	0.35	0.23	0.73
16	0.2	0.35	0.30	0.93
18	0.2	0.35	0.50	0.97
20	0.2	0.35	0.57	0.97
22	0.2	0.35	0.63	1
24	0.2	0.35	0.63	1
26	0.2	0.35	0.63	1
28	0.2	0.35	0.63	1
30	0.2	0.35	0.63	1

include all possible solutions to the user query. In another words, it must be lengthy enough to include all documents relevant to the query entered. The relevant number of documents per queries passed to the GA unit varies between 2 and 105, as shown in Figure 3-15. Bearing in mind that the criterion of selecting the document is that it must have at least one keyword. Hence the number of documents that satisfy this condition is much higher than the number of relevant documents. Therefore, there is a need to examine the probability of success for the chromosome length in order to select the chromosome length suitable for the GA unit of IRUGA.

In order to determine the best chromosome length, an experiment similar to the one conducted in the previous section is performed here. The probability of success is applied here by fixing the population size at 125 which is the optimal one obtained earlier, and performing several experiments on 4 queries for chromosome lengths $cl \in \{50, 75, 100, 125\}$. The results obtained are illustrated in Figures 3-16 to 3-19 and Table 3-5.

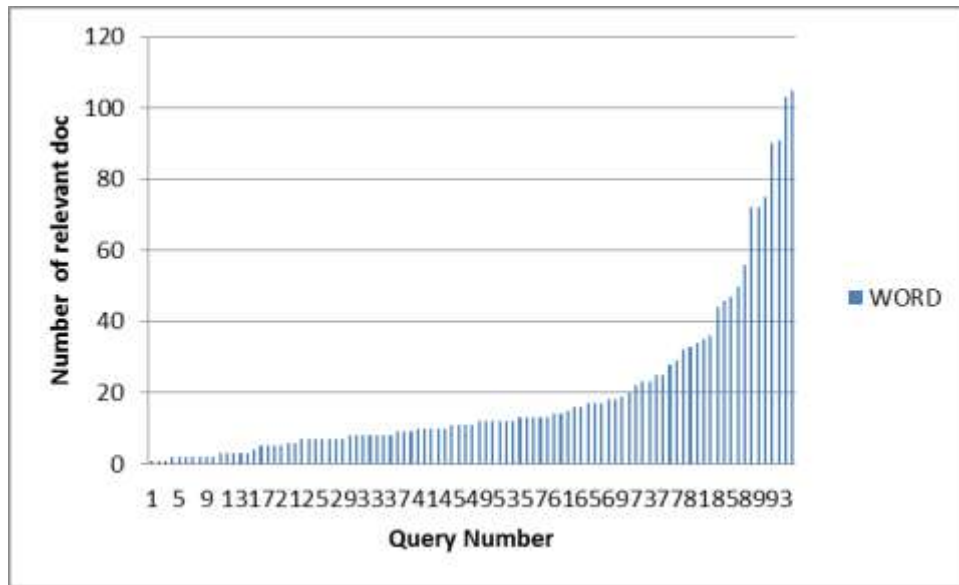


Figure 3-18: Number of relevant documents per each query number.

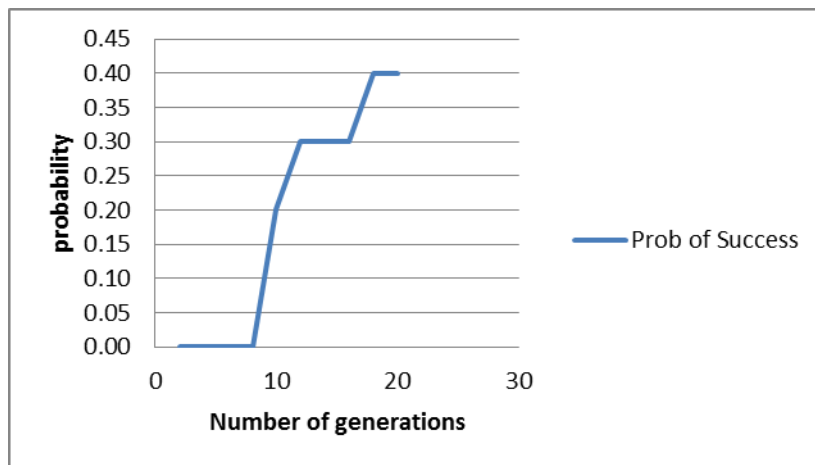


Figure 3-19: Probability of success for population size =125 and chromosome length $cl = 50$

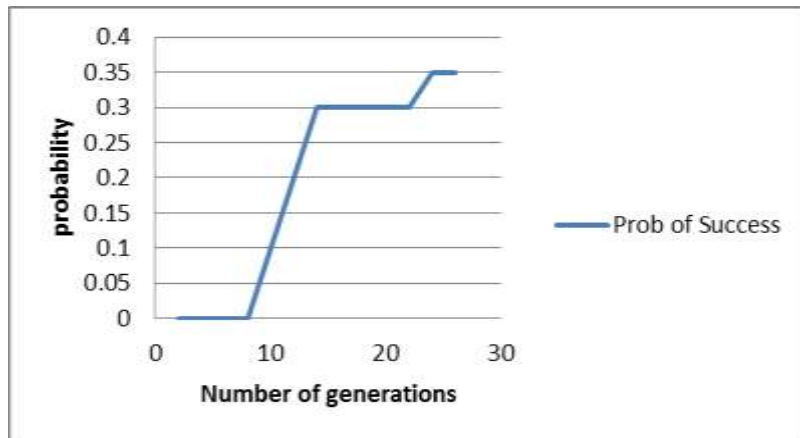


Figure 3-20: Probability of success for population size =125 and chromosome length $cl = 75$

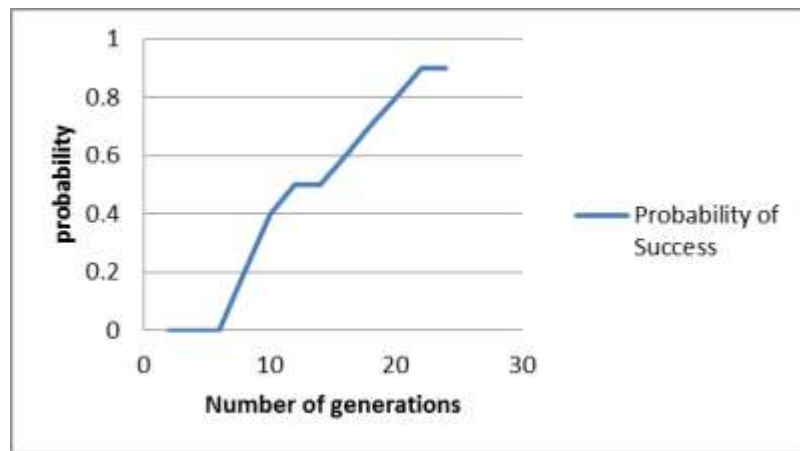


Figure 3-21: Probability of success for population size =125 and chromosome length $cl = 100$

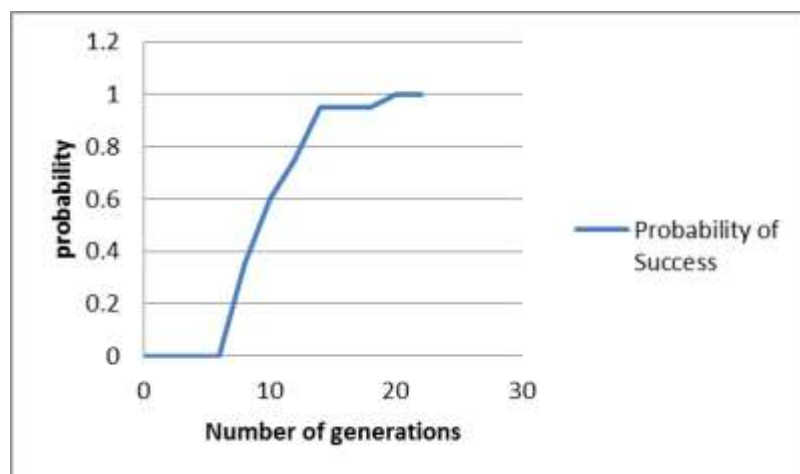


Figure 3-22: Probability of success for population size =125 and chromosome length $cl = 125$

These figures show that at generation 10, the probability of success in the GA unit of IRUGA system is equal to 10%, 10%, 40% and 60% when $cl = 50, 75, 100, 125$ respectively, while this probability of successes for finding the optimal solution increases to 20%, 35%, 90% and 100% at generation 21. The results obtained illustrate that when cl value is equal to 125, the GA unit of IRUGA system is able to produce the optimal test results. Hence, the adopted chromosome length for the GA unit of IRUGA is 125.

Table 3-5: The probability of success of IRUGA for population size = 125 and different chromosome length.

Generation number	chromosome length =50	chromosome length =75	chromosome length =100	chromosome length =125
2	0	0	0	0
4	0	0	0	0
6	0	0	0	0
8	0	0	0.2	0.35
10	0.1	0.1	0.4	0.6
12	0.15	0.2	0.5	0.75
14	0.15	0.3	0.5	0.95
16	0.15	0.3	0.6	0.95
18	0.2	0.3	0.7	0.95
20	0.2	0.3	0.8	1
22	0.2	0.3	0.9	1
24	0.2	0.35	0.9	1
26	0.2	0.35	0.9	1
28	0.2	0.35	0.9	1

3.7.3 Crossover Rate

Crossover rate represents the probability of applying the crossover during the creation of the next generation. In (Minaei-Bidgoli and Punch, 2003; Picarougne, Monmarche, Oliver, and Venturini, 2002b; Radwan et al 2006; Aly, 2007; Cutler et al, 1999; Martín-Bautista and Vila, 1998; Radwan et al,2006) this rate is between 0.6 and 1. The idea behind making crossover rate less than one is to allow some random parents to be copied to the next generation without change. However, if this process is modified such that only the best individual is copied unchanged to the next generation, which is called *elitism*, then there is no need to reduce the rate of crossover. Consequently, the crossover rate in the GA unit of IRUGA is set to one. This means that the crossover is performed each time the offspring is generated and the elitism will improve the GA unit performance where the best individuals

are always maintained and driven to the next generation.

3.7.4 Mutation Rate

The mutation rate represents the probability of applying the mutation during the creation of the next generation. The mutation rates used in similar studies range between 0.001 (Beasley et al, 1993a) to 0.7 (Radwan et al 2006). However, the mutation is more useful when search space is large (Milutinovic, Cvetkovic and Mirkovic, 2000). Hence, the mutation rate applied in the GA unit of IRUGA is 0.7. The reason behind using this high rate is to increase the chance of modifying the offspring by injecting it with better genes. In addition, it increases the speed of convergence since many species in the search space are parsed. Moreover, high mutation rate has an effect on avoiding convergence into local optima (Minaei-Bidgoli and Punch, 2003).

To justify this conclusion, an experiment will be conducted in Chapter 5 using four mutation rates in order to compare their performance in terms of precision and recall.

3.7.5 Maximum Number of Generations

For each run, the maximum number of generations reached is 35. The experimental results in Section 4.3 show that 35 generations of evolution is enough because all the fitness values converge before the fiftieth generation. Figure 4-9 shows that the system reaches the probability of success of 1 at generation number 22. This result represents the average of 10 runs. To be on the safe side and not to force the GA to divert to unexpected results, the maximum number of generations *max_g* can be set to a somehow higher value. Hence it is set to 50.

3.7.6 Termination Criteria

GA is an iterative process where each iteration is called a generation. The iterative process of the GA unit of IRUGA will be stopped by one of the following two termination criteria: the first criterion is that there is no improvement on the current generation performance compared with the previous generation. The generation performance is measured as the summation of fitness values of all individuals within the generation. However, the GA unit of IRUGA stops if the difference between two consecutive generations is less than the

predefined threshold, i.e., $E(G_i) - E(G_{i-1}) < threshold$. To have a better solution, this threshold is set to a very small value compared to the possible chromosome fitness value. In fact, the fitness value of the population in the GA unit depends on two factors, namely, the chromosome length cl and the population size ps .

According to the description of the fitness functions described in Chapter 3, the maximum fitness value for the document is almost one. Hence, the maximum fitness value of the generation is $cl \times ps \times 1$. In the GA unit of IRUGA, the threshold value is set to be very small compared to this value; therefore it is set to *one*, which is much smaller than $cl \times ps$. That means, if the difference in the fitness value between the current generations and the previous generations is less than one, the system halts.

The second termination criterion is when the predefined maximum number of generations is reached. In both cases, the solution is an individual of the last generation that has the highest fitness value.

3.7.7 Summary of setting up the parameters of GA unit

The parameters of the GA unit of IRUGA are summarized in Table 3-6. In this table it is shown that the population size of each generation is set to 125. The chromosome length is also set to 125, while the crossover rate is set to 1 which means that the crossover is performed for each process of creating the offspring. Moreover, the elitism is done for the best chromosome only to be passed to the next generation without passing through the crossover process. However the mutation rate is set to 0.7. Finally, the maximum number of generations is set to 50. These values are set based on the experiments conducted earlier in this section.

Table 3-6: Parameter setting of IRUGA

Parameter Description	Value
Population size	Fixed at 125
Maximum number of generations	50
Chromosome length	125
Crossover rate	1
The number of best individuals copied to the next generation (Elitism)	1
Mutation rate	0.7

3.8 The Description of IRUGA Comparisons

To prove the effectiveness of the proposed GA techniques adopted for IRUGA, a set of comparisons is conducted. This section is going to explain the comparison methods which will highlight the effect of the proposed technique for each operator.

When examining the effect of a particular technique, the GA parameters and other operators are kept unchanged.

Each comparison method is performed by fixing all operators and parameters except the one under consideration, and studying the performance of the variations applied to it.

3.8.1 Comparison Methods of the Enhanced Inverted Index

As mentioned in Section 3.3, EII is implemented using the Oracle database, which has the ability to index a large number of documents and allow their fast retrieval. Hence, the storage space required to store the index of 8344 documents using EII will be compared with the expected storage space occupied by the vector space, bearing in mind that IRUGA requires additional data to be stored for each term beside the indexed terms and the list of documents referencing it.

In addition, the time required to retrieve the document will be monitored. This time will be calculated from the moment the user enters the query until the document list is retrieved and displayed to the user.

3.8.2 Comparison Methods of Evaluating the GA unit of IRUGA

The GA unit of IRUGA is composed of a set of operators and controlled by a set of parameters.

The GA unit of IRUGA is evaluated in terms of recall and precision through a set of comparison methods. These comparison methods are described in the following subsections.

3.8.2.1 Comparison Methods of the Initial Generation Creation

The first generation of the GA unit is created using the random selection technique with selective criteria as explained in Section 3.4.1.

There are two steps that must be considered here. The first one is to check IRUGA's performance in terms of recall and precision. The second one is to compare the speed of convergence between the two IRUGA models, namely, the *selective random selection* and the *pure random selection*.

3.8.2.2 Comparison Methods of the Selection Operator

The selection method used in the GA unit of IRUGA is the *modified binary tournament* selection. It is implemented such that two individuals are chosen at random from the population; random number r is then generated between 0 and 1. If $r < k$, (where k is a parameter set to 0.75 to favour the fitter individual), then the fitter individual is selected (Chen and Dhar, 1991), otherwise the weaker one is selected. This technique is denoted as *parent selection -75*. The idea behind this technique is to allow weak individuals to participate in the solution since they may contain good genes (documents) that have a particular degree of relevance and can increase the performance at some stages. Therefore, this technique gives a chance for these documents to be selected. The common technique applied in the tournament selection is the one that favours the fitter one with probability equal to one, and denoted as *parent selection -100*.

In this experiment, the comparison will be applied between *parent selection -100* and *parent selection -75* in terms of speed of convergence, recall and precision.

3.8.2.3 Comparison Methods of the Crossover Operator

Crossover is one of the main GA operators which are used to produce a new generation. It is the process of producing offspring from two parents. The proposed crossover operator is the *hybrid crossover* technique as explained in Section 3.5.2.

The comparison methods that will be used in the GA unit of IRUGA carried out in this regard have three disciplines. The first one shows that the hybrid crossover technique performs better than the two-point crossover in terms of recall and precision. The second experiment is to show that the hybrid crossover technique performs better in terms of recall and precision than the normal crossover which produces two offspring. The last experiment is to demonstrate that the hybrid crossover technique performs better than the classical one-point crossover which is applied to a non-ordered crossover.

3.8.2.4 Comparison Methods of the Fitness Functions

In the GA unit of IRUGA, two fitness functions are developed. One uses both local and global factors and called the *multi-terminal fitness function*. The second fitness function uses only local factors and called the *term proximity fitness function*. Details of these fitness functions are explained in Section 3.4.4.

This experiment examines the performance of the TPF fitness functions in comparison with two other well-known fitness functions in this domain, which are the Okapi-BM25 (Noreault, McGill, & Koll, 1980) and the Bayesian inference network model (Kim and Zhang, 2003) fitness functions. According to Zhang (2009), the state-of-the-art retrieval function used in information retrieval is the Okapi-BM25. Moreover, this function consistently performs very well in TREC competitions (Fan et al, 2004; Manning, Raghavan and Schütze, 2009, p. 234). This function is based on two main factors, namely, term frequency and document length. Both The Bayesian inference network model and OKAPI-BM24 are based on the probabilistic models in document retrieval.

In this experiment, all the parameters and operators of the GA unit of IRUGA are fixed, and the performance will be monitored for each fitness function and will be examined against the recall and precision achieved by each one of these functions.

3.8.2.5 Mutation Comparison Methods

The mutation concept used here is the basic one which selects one gene randomly and replaces it with another one chosen randomly from the search space with the new gene, which has a better performance than the replaced one. Moreover, the mutation rate adopted here is 0.7, which is the maximum rate used in similar studies, and it varies between 0.001 and 0.7.

Therefore, the experiment conducted here is to demonstrate that the GA unit of IRUGA is expected to have better performance when using a mutation rate of 0.7. This is achieved by performing a comparison between different mutation probabilities. The experiment done for this purpose will be for the mutation probabilities 0.001, 0.1, and 0.2, in addition to the 0.7 which is adopted in the GA unit of IRUGA.

3.9 Summary

This chapter has described in detail the design of IRUGA. It starts by the pre-processing step. In this process an enhanced inverted index is created using Oracle database and enhanced to include HTML tag weight for each term. This index stores for each indexed document a list of terms along with their offset, HTML weight and sentence index. The six most frequent HTML tags are adopted in this model and assigned a weight from 1 for the body text to a weight of 6 for the title, as it best describes the document. Other HTML tags included are header, anchor, bold and italic.

The proposed IRUGA is characterized by the following:

1. IRUGA is a GA-based IR system.
2. It is applied to HTML documents.
3. The HTML documents are represented using an advanced inverted index model.
4. It uses a set of GA operators that are capable of producing high quality results. These operators are further enhanced to meet the IRUGA requirements.
5. It evaluates documents using an innovative fitness function: TPFf.
6. IRUGA is capable of evaluating documents in the search space based on the number of referenced query keywords.
7. It ranks the results based on the user query.

After creating the index, the user can enter his query which consists of several words. These words are considered as keywords. The GA unit of IRUGA starts by creating initial population randomly from the search space such that these individuals must have at least one keyword of the user query. Next generations are reproduced using three operators, named: selection, crossover and mutation and each individual is evaluated using the fitness function.

The selection method applied in this framework is the binary tournament selection in which two individuals are selected randomly from the population, and then the one with higher

fitness value is nominated as first parent with probability equal to one. The second parent is selected in the same manner from another two randomly selected individuals.

Among several crossover techniques developed in this domain, a special crossover technique is proposed for the GA unit of IRUGA which is the hybrid crossover technique that combines one-point crossover, ordered crossover and fusion crossover. One-point crossover was selected since it is more suitable than the n-point crossover ($n > 1$) for chromosomes with ordered genes. For the IRUGA to retrieve the related document at high position and high rank, the ordered crossover is applied beside a one-point crossover. Finally, to reduce the number of iterations to get the best generation, a fusion crossover method is applied where one offspring is produced from two parents with best genes inherited from both parents.

Simple mutation technique is applied in the GA of IRUGA. The main reason behind selecting this technique is that it requires less processing load where mutated genes are selected randomly and replaced with new randomly generated ones. Two things need to be considered before replacing the old gene in the mutated chromosome. The first one is that it must preserve the uniqueness of genes within the chromosome and the second one is that it must have a better fitness value than the replaced one.

Two fitness functions are developed to evaluate the individuals. The first one is called multi-terminal fitness function which uses a combination of local and global factors which are categorized as statistical, formatting, and semantic factors. The best documents retrieved by this function are those which contain all query keywords, and have a number of unique keywords which are almost the same as the number of unique terms in the document. Moreover, this document has a very small frequency of the query keywords compared to the frequency of all terms in the set, and the number of documents referencing the query keywords is very small compared to the total number of documents in the space. Finally, the HTML tag *weight* is very high.

The second fitness function developed that features the GA unit of IRUGA is called the Term Proximity Fitness Function. It uses only local factors. And these factors include statistical, formatting, and semantic factors. The best documents returned by this formula are those which have all keywords of the query, in the same sequence as that of the query,

and adjacent to each other, and appear at the very beginning of the document. Moreover, they appear in important HTML tags and have high frequency among the document.

This chapter also described the environment configuration and the parameter settings of IRUGA in addition to the experiments that need to be performed in order to test the performance of the GA units of IRUGA. The system configuration includes the platform specifications and the programming language used. The parameter setting includes population size, chromosome length, crossover probability, mutation probability, and termination criteria. The values of population size and chromosome length are selected based on the empirical studies so that these values provide the highest probability of success. On the other hand, the parameter values of crossover probability and mutation probability are set to one and 0.7 respectively. Justifications of these values are presented in Sections 3.7.3 and 3.7.4. The maximum number of generations is specified based on the experiments performed. The termination criteria of the GA unit of IRUGA are one of two choices: either the performance of the current generation compared to the previous generation is not improved, or the maximum number of generations is achieved.

The comparison methods to be performed in order to test the GA unit of IRUGA performance compared with existing techniques used in other GA systems are described in this chapter. These comparison methods examine the performance of IRUGA based on the enhanced techniques described in this chapter that need to be applied to each operator. This section has highlighted the layout of the comparison methods that need to be performed. For each technique used to implement an operator there is a set of experiments that needs to be executed by fixing all operators and parameters except the one under consideration. The results of evaluating these techniques are presented and analyzed in the next chapter.

Chapter Four: Experiments and Results Of IRUGA

4.1 Introduction

As explained in previous chapters that IRUGA is GA –based IR system that applies the GA concept to retrieve the relevant documents based on user query. The design of IRUGA and its units has been explored in Chapter 3. Furthermore, the environment setting of IRUGA and the parameter setup of the GA unit are defined in Chapter 4. The current chapter aims to evaluate the proposed techniques in terms of several well known measures applied to IR systems.

The first part of this chapter describes the document set as well as the queries used to examine IRUGA. The second part describes the measures used to evaluate the performance of IRUGA. These measures are recall at rank N, precision at rank N and precision at recall M, where N is multiples of 10 and M is multiples of 10%. The storage space required to store the indexed documents using the enhanced inverted index (EII) is compared with the space required by the vector space model. The third part of this chapter examines the performances of all operators of the GA unit of IRUGA against the proposed techniques explained in Chapter 3.

The proposed experiments are performed by fixing all the parameters and fixing the techniques of all operators except the one under investigation. The results are averaged to produce the final figures. These figures are plotted graphically and/or presented in tabular form if the figures are close and cannot be analyzed graphically.

IRUGA aims ultimately to produce an IR system that is able to retrieve the relevant documents based on the user query. These documents must satisfy two criteria. The first criterion is that the obtained results must have high recall, i.e. retrieving from the search space as much relevant documents to the user query as possible. The second criterion is that the results must have high precision, i.e. the least possible irrelevant documents from the

search space.

This chapter is organized as follows: Section 4.2 describes the document set used, while Section 4.3 describes the queries applied and Section 4.4 explains the relativity measures adopted to test the performance of IRUGA. Section 4.5 is the main section where it shows the results of each operator of IRUGA. Finally, Section 4.6 concludes this chapter.

4.2 Document Set Description

In similar studies, researchers tend to use ready-made data sets which use vector space indexing models such as TREC and ACAM data sets. These sets include documents, vector space index, queries and their results. However, these sets are not suitable for IRUGA because of the indexing model on the one hand, and due to the additional data that need to be included in the index which is not supported by these data sets on the other hand.

The document set or search space of IRUGA is a set of HTML web documents. This set is the Carnegie Mellon University data set (WebKB). It is a set of HTML documents from the departments of computer science at various universities collected in January 1997 by the World Wide Knowledge Base project of the CMU text learning group. It consists of 8284 documents (The 4 Universities Data Set, 1998) and used by several researches (Craven, et al., 1998; Dong et al, 2008). This set consists of seven categories, named: course, department, faculty, project, staff, student and others, in addition to another 60 web documents downloaded from the Web by passing different keywords to the Google search engine. Hence, the total number of HTML documents in the set is 8344. Table 5-1 shows the categories of the document set as well as the number of documents in each category, and

Table 4-1: Categories of documents used to test IRUGA

Category	No. of Documents
Courses	927
Department	183
Faculty	1130
Project	504
Staff	137
Students	1639
Others	3764
Random web pages	60
Total	8344

Table 4-2: Statistics of the test collection used in IRUGA

Parameter Name	Value
Number of documents	8344
Number of queries	100
Number of unique indexed terms	128213
Average number of terms by query	2.69
Average number of relevant documents by query	16.82278
Average number of indexed terms by document	410.2792

Table 4-2 shows some statistics for the documents and queries used to test the IRUGA. Using the document set made up of 8344 documents is expected to be reasonable to analyze IRUGA since this size is in the range of document size used in similar researches. In the literature, the data set used to test most GA-based IR systems is CISI. This data set consists of 1460 documents and was tested against 76 to 112 queries. The data sets used in such researches are summarized in Table 4-3 which shows that the size of these sets varies from 300 to 247,491 documents. By referring to this table, it is found that (Cutler, 1999; Radwan et al 2006; Billhardt et al, 2002; Aly, 2007; Vrajitoru, 1990) are using a data set ranging in size between 3040 and 3204, which is bigger than CISC. This gives an indication that the data set size used in this study is within the acceptable range.

Table 4-3: Data sets used in similar IR researches.

Data set	No. of Documents	No of queries	Reference
Downloaded pages from the standard four search engines (Yahoo, Google, AltaVista, MSN)	300	10	(Marghny and Ali, 2005)
Medline	1033	30	(Billhardt et al, 2002; Cummins and O’Riordan, 2006; Salton and Buckley, 1990)
CRAN	1398	225	(Billhardt et al, 2002; Salton and Buckley, 1990)
Cranfield	1400	225	(Cummins and O’Riordan, 2006)
CISI	1460	76-112	(Radwan et al 2006; Vrajitoru, 1997; Billhardt et al, 2002; Cummins and O’Riordan, 2006; Aly, 2007; Salton and Buckley, 1990)
Binghamton University at the end of 1996	3040	10	(Cutler et al, 1999)
CACM	3204	52-64	(Radwan et al 2006; Billhardt et al, 2002; Aly, 2007; Vrajitoru, 1997; Salton and

			Buckley, 1990)
WebKB	8284	100	(Craven, et al., 1998; Donget al , 2008)
NPL	11,429	100	(Radwan et al 2006; Aly, 2007; Cummins and O’Riordan, 2006; Salton and Buckley, 1990)
INSPEC	12684	84	(Salton and Buckley, 1990)
OHSU88	70,825	61	(Cummins and O’Riordan, 2006)
OHSU89	74,869	63	(Cummins and O’Riordan, 2006)
OHSU90-91	148,162	63	(Cummins and O’Riordan, 2006)
TREC	247,491	-	(Kim and Zhang, 2003; Kim and Zhang, 2000)

4.3 IRUGA Queries Setting

IRUGA system is tested on 100 queries prepared specially for this purpose. These queries vary in length from two to five words. The number of queries used in similar studies is shown in Table 4-3. By comparing the number of the queries to the 100 queries used in IRUGA, it is found that their number is quite reasonable to test the effectiveness of IRUGA.

In literature, researchers tend to use a predefined set of queries where the relevant documents are known. However, due to certain circumstances, such sets are not available for this research. Therefore, a new set of queries has been created. They are created such that some of them have a small number of relevant documents and some have a large number of relevant documents. These queries are encoded as vector $Q = \{q_1, q_2, \dots, q_n\}$, where $n > 1$ and q_i represents the term within the query. Only one condition is applied to the created queries. This condition is that the query vector length must be greater than one in order to make the query meaningful. In addition, it limits the number of relevant documents. A list of these queries associated with relevant statistics is presented in Appendix C.

In order to judge how relevant each document is to the created queries, three steps are applied. The first step is to find the documents that reference all query keywords. The second step is to filter those documents to pick the ones that allow these keywords to appear in a distance equal to the number of query keywords or less by one. In other words, these keywords appear adjacent at least once within the document. The last step is to examine the filtered documents visually in order to be considered for relevance.

4.4 Evaluation Measures

The results of the proposed system are evaluated by using precision and recall measures.

Precision is defined as the percentage of relevant retrieved documents to the total number of retrieved documents, while recall is defined as the percentage of relevant retrieved documents to the total number of relevant documents (Desjardins, Godin, and Proulx, 2005; Horng and Yeh, 2000).

One of the most popular measures used to evaluate the IR systems is called average precision-recall measure (Pathak, Gordon and Fan, 2000) where it is used in (Kim and Zhang, 2003; Radwan et al 2006; Kim and Zhang, 2000; Aly, 2007; Horng and Yeh, 2000; Desjardins, Godin, and Proulx, 2005). It measures the precision at multiples of 10% of the total relevant retrieved documents for the given query. In other words, if the query has 100 relevant documents, then this measure will evaluate the precision when retrieving 10, 20, 30,..., 100 relevant documents. Therefore, this measure evaluates the system in terms of percentage of the total relevant documents.

In addition to the average precision-recall measure, two common measures are used to evaluate such systems. These measures are: Precision at Rank N ($P@N$) and Recall at Rank N ($R@N$), where N is multiple of 10 (Kim and Zhang, 2003; Cho and Richards, 2004). Rank N here means the top N ranked documents of the retrieved documents. In this method, the retrieved documents are ranked in descending order based on the fitness value and the average of precision and recall are calculated. Therefore, this measure evaluates the system based on the number of the total retrieved documents.

When the maximum value of N is 100, this measure is called 11-point average precision (Kim and Zhang, 2000) and it is widely used to evaluate IR models (Cutler et al, 1999), since it measures the performance at the points 0, 10, 20, 30 up to 100 top ranked retrieved documents, where point 0 means first retrieved document. However, some authors use a smaller value for N, such as Carlberger et al (2001) who evaluate the average precision-recall for $N=10$ only, and Vrajitoru (1998) uses $N \in \{5, 10, 15, 20, 25, 30\}$.

In addition to $P@N$ and $R@N$ measures, there are several measures used to evaluate the IR system considering the documents in terms of relevance and retrieval factors. These factors are shown in Table 4-4 (Horng and Yeh, 2000).

Table 4-4: The parameters used in the measurement of document retrieval (Horng and Yeh, 2000)

Number of documents	Relevant	Non-relevant
Retrieve	a	b
Not retrieved	c	d

Examples of the evaluation measures that use these factors are: recall ratio (R) = $a/(a+c)$, precision ratio (P) = $a/(a+b)$, fallout ratio (N) = $b/(b+d)$, and F1 measure = $2a/(2a+b+c)$ (Horng and Yeh, 2000). However, these measures require a threshold to classify the document as relevant or not. This threshold must be chosen carefully so as not to retrieve irrelevant document if the threshold is low and not to miss relevant documents if the threshold is high (Horng and Yeh, 2000, p.745-746).

In addition to these measures, there is another measure used to obtain the average fitness per generation (Pathak, Gordon and Fan, 2000; Kim and Zhang, 2003). However, this measure better assesses the improvement of GA through several generations, which compares different generations in terms of the total value of fitness. Therefore, it is not consistent with the objective of IRUGA.

In IRUGA, three measures are adopted to evaluate the proposed algorithm. The first one is the average precision-recall. It is chosen since it is a common measure in this field (Pathak, Gordon and Fan, 2000). The second measure is the *precision at rank N* ($P@RankN$). The third one is the *recall at rank N* ($R@RankN$). The last two measures are adopted as they simulate the behaviour of the user toward the results of the search engine, where the user normally measures the performance of the search mechanism by concentrating on the results appearing in the first few pages, and each page normally retrieves 10 documents. N can be considered as the number of results per page that is displayed to the user. In addition to $P@RankN$ measure, precision at 11-points is also considered due to its popularity in evaluating IR models (Yeh et al, 2007; A. Aly, 2007; Desjardins, Godin, and Proulx, 20; Kushchu, 2005; Kim and Zhang, 2000).

In addition to these measures a new measure can be considered to further evaluate the performance of IRUGA. This measure is the convergence speed of the GA unit. In this

measure, the number of generations for each technique used in the GA unit of IRUGA is recorded and is then compared with the existing techniques to measure how fast the proposed techniques converge. This gives a clue of how fast this technique will present the results to the user. In fact, this measure was not tackled by researchers explicitly, but need to be introduced in this work as a new measure.

These measures will be applied to each technique of the GA unit of IRUGA mentioned in the previous chapter whenever applicable, and each measure will be presented graphically in a separate diagram to compare different methods in terms of recall percentage or precision percentage.

4.5 Experimental Baseline

The baseline or benchmark of the experiments to be performed is based on two parts according the literature. The first considers the document representation. It is found that many researchers use VSM as baseline for the document representation such as (Vrajitoru, 1998; Billhardt, Borrajo, and Maojo, 2002). Because drawbacks of VSM are already mentioned in Section 2.2.3, this baseline is omitted from this study.

What ready concerns here is the evaluation of the document and determining the degree of relativity of it to the user query, Some researches such as (Yeh et al, 2007) have adopted BM25 as baseline for thier expiremnts. While (Jones, Walker, and Robertson, 2000; Yoshioka and Haraguchi, 2005; Yoshioka and Haraguchi, 2005; Pohl, Zobel, and Moffat, 2010) use the OKAPI functions as a baseline and (Lops et al, 2012) adopt OKAPI as a scoring function for their model because it is still considered as one of the state-of-the-art retrieval model. However, it is been shown that yet there is a better evaluation function that can be used as a baseline (Fan, Fox, Pathak, and Wu, 2004) which is OKAPI-BM25. Advantages of this evaluation fucntions are discussed in Sections 2.4.4.1 and 3.8.2.4. Hence the baseline for this thesis will be the OKAPI-BM25 evaluation function as it is been adopted by many researches .

4.6 Testing the Performance of the Enhanced Inverted Index

The expectation of the enhanced inverted index is to use less space than the vector space and

Latent Symantec index model. By analyzing the document collection, it is found that it consists of 128,213 unique words and 8344 documents. The index needs to store several kinds of information for each word. The space required to store one kind only is to be considered using the vector space model. This kind of information is the frequency of each term. When using two bytes to store each entry, then this will require $128,213 \text{ words} \times 8344 \text{ documents} \times 2 \text{ bytes}$ which is equivalent to about 20 GB of storage.

In fact, when using the inverted index, the storage space required to store the whole index with all needed details of each document consumes only 100 MB. This implies that the enhanced inverted index saves almost 99.5% of storage space.

4.7 Testing the Operators' Performance of the GA unit of IRUGA

As stated in Chapter 3, the operators of the GA unit of IRUGA are implemented using specific techniques. The results of evaluating these techniques are presented here, starting with the technique of creating initial generation.

4.7.1 Convergence Speed of IRUGA Operators

This section examines the convergence speed of the GA unit of IRUGA. It is evaluated by considering the last generation number created by each technique for each operator. The average number of generations is obtained by running each technique once on the 100 queries. Then the last generation is recorded for each one of these 100 queries. After that, the average of these records is taken to represent the average convergence for the technique under consideration.

These averages are presented in Table 4-5. The techniques appearing in bold in the table are the techniques adopted for the GA unit of IRUGA. The figures represent the last generation number of each technique. The numbers in bold represent the corresponding results of the GA unit of IRUGA.

There are several observations that can be drawn after examining the results in this table.

- The first observation is that creating initial generation using the selective criteria mentioned above leads to faster convergence since the *selective random selection* technique converges at the 22nd generation, whereas the *pure random*

selection technique converges at generation 25. This implies that the proposed selection criterion speeds up IRUGA by 12%.

Table 4-5: The average convergence of each technique.

Operator Name	Technique Name	Average Convergence
Initial selection	Initial generation with selective criterion (selective random selection)	22.26
	Random selection of initial generation (pure random selection)	24.96
Parent selection	binary tournament selection which always favours better parent (Parent Selection-100)	22.26
	binary tournament selection which favours better parent with $P \leq 0.75$ ((Parent Selection-75)	22.90
Crossover method	Hybrid crossover	22.26
	Non-ordered crossover representation	28
	One-point crossover and producing two offspring	43.40
	Two -point crossover producing one offspring	13.65
Mutation rate	70% -MUTE70	22.26
	0.1% - MUTE001	22.13
	20% - MUTE20	22.56
	10% - MUTE10	22.30
Fitness function	Proximity term fitness function	22.26
	Okapi-BM25	41.66
	Bayesian inference model	23.58

- The second observation is that the binary tournament selection using the *parent selection-100* technique has the same performance as the *parent selection-75* technique except for a minor improvement where the former one is faster by only one generation.
- The speed of convergence differs widely from one technique to another. It is noted that the fastest convergence is achieved by the 2-point crossover technique. This is because the genes between the 2 cross points have similar performance, since the good and bad genes are mixed together. Hence, exchanging them will not affect the overall chromosome performance. That means: the offspring will have almost the same or very close performance as the parents. Therefore, GA converges fast. However, the performance of this technique in terms of recall and precision is very low and comes in the third position when compared with other crossover techniques illustrated in Section 5.6.4.

- The slowest crossover technique is the one that applies a one-point crossover on ordered chromosomes and produces two offspring. The reason is that the good genes are bouncing between the two offspring, which delays the convergence.
- The speed of convergence is not dependent on the mutation rate, since from rate of 0.001 to 0.7, all queries are converging at 22nd generation.

4.7.2 The comparison results of the Initial Generation Creation Techniques

It is mentioned in Section 3.4.1 that there are several ways of creating the initial generation. The most popular one is to select the population randomly from the search space. This technique is called *pure random selection*. On the other hand, selecting the initial population with specific criterion – which is called *selective random selection* - may enhance the performance in terms of the speed of convergence and the percentage of recall and precision. In the GA unit of IRUGA, the selection criterion adopted is to select the documents that reference at least one keyword from the user query.

This experiment will examine the speed of convergence of the GA unit of IRUGA between these two methods. Table 4-5 (page 149) shows the average convergence which represents the last generation of each mentioned technique. Note that this table represents the different techniques applied in the GA unit of IRUGA by using the parameters that are set in the previous chapter. The results obtained for this experiment show that creating the initial population using the above mentioned selective criteria leads to faster convergence since *selective random selection* technique produces the results at the 22nd generation, whereas *pure random selection* technique produces the results at generation 25. This implies that the proposed selection criterion speeds up the system by 12%.

Looking at the performance of these selective techniques in terms of the precision measure, it is found that the precision at the top 10 retrieved documents reaches 85% when applying the *selective random* selection while *pure random* selection achieves only 60%. It is noted also that the average precision for the *selective random* selection is 0.49 while the average of pure selection is 0.31 showing that the former technique achieved enhancement of 101.94%. These results are illustrated in Figure 4-1 and Table 4-6.

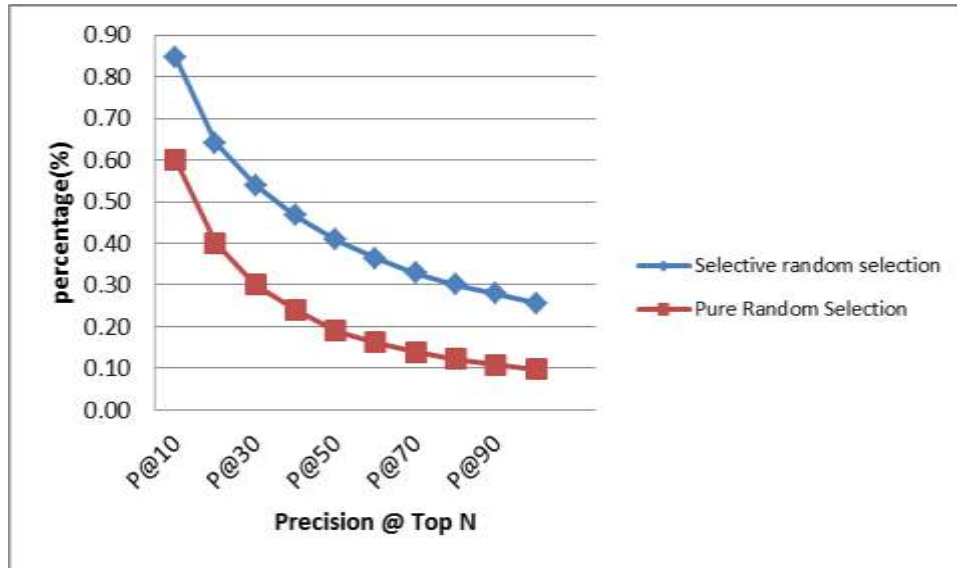


Figure 4-1: Comparison of P@N for different selection techniques

The next measure to be investigated is the recall @ N. As illustrated in Figure 4-2, *selective random selection* retrieved 88% of the relevant documents at the top 50 retrieved documents, in the time *pure random selection* retrieved only 42%. Moreover, the GA unit of IRUGA which uses the selective random selection retrieved 94% of the total relevant documents at the top 100 retrieved documents, whereas only 44% of total relevant documents appear within the top 100 retrieved documents when using the pure random

Table 4-6: The P@N enhancement percentage for different selection techniques

Selection technique	Selective random selection	Pure Random Selection	% of improvement
P@0	1.00	1.00	0
P@10	0.85	0.60	40.87
P@20	0.64	0.40	60.10
P@30	0.54	0.30	79.90
P@40	0.47	0.24	94.35
P@50	0.41	0.19	115.14
P@60	0.36	0.16	125.16
P@70	0.33	0.14	136.43
P@80	0.30	0.12	147.05
P@90	0.28	0.11	158.57
P@100	0.26	0.10	163.80
Average	0.49	0.31	101.94

selection technique. This implies that using the selective random selection enhances the recall @ N measure by 110.02%. The recall @ N retrieved for both techniques along with percentage of enhancements is included in Table 4-7.

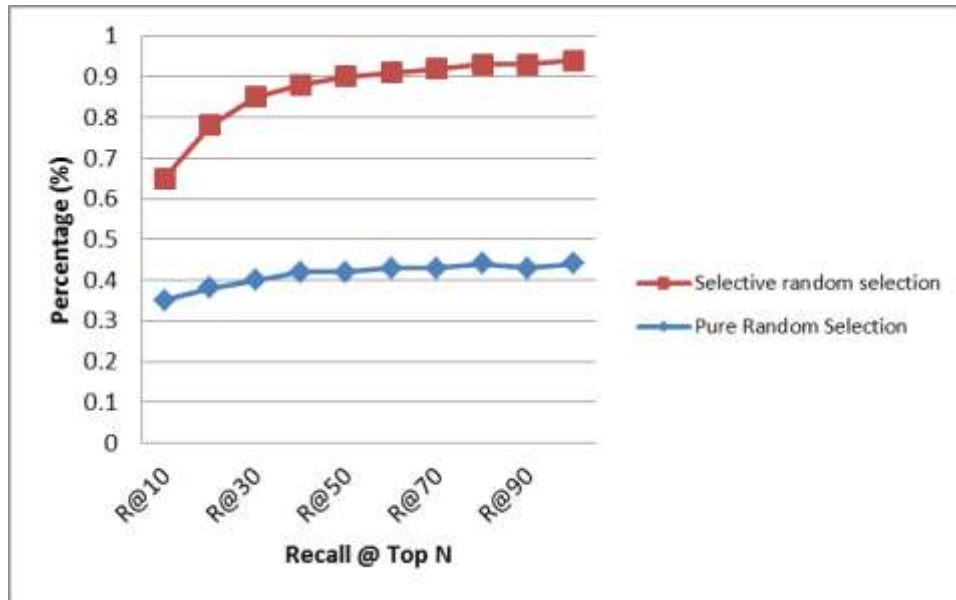


Figure 4-2: Comparison of R@N for different selection techniques

The third measure to be demonstrated here is the precision @ recall. In this measure, the performance is examined: for example, when retrieving 10% of total relevant documents, what will be the precision within this window of 10%. The results of this measure are shown

Table 4-7: The R@ N enhancement percentage for different selection techniques

Selection technique	Selective random selection	Pure Random Selection	% of improvement
R@10	0.65	0.35	86.08
R@20	0.78	0.38	103.97
R@30	0.85	0.40	114.13
R@40	0.88	0.42	110.56
R@50	0.90	0.42	112.88
R@60	0.91	0.43	113.68
R@70	0.92	0.43	113.68
R@80	0.93	0.44	113.68
R@90	0.93	0.43	115.76
R@100	0.94	0.44	115.76
Average	0.87	0.41	110.02

in Figure 4-3 and Table 4-8. Selective random selection and pure random selection techniques start by a precision of 100% and 98% when retrieving 10% of total relevant documents respectively. However, selective random selection technique achieves a maximum of 98% when retrieving 50%, while pure random selection achieves a maximum of 67% when retrieving 50%. Moreover, the precision of the first technique drops to 87% when the number of retrieved documents is equal to the number of relevant documents. On the other hand, the precision for pure random selection drops until reaching 36% when retrieving a number of documents equal to the number of relevant documents. The comparison in the case of precision versus recall is sometimes done by comparing the number of retrieved documents to the total number of relevant documents instead of the actual retrieved documents. Since the retrieved documents are arranged in a descending order, this implies that the precision value at 100% recall reflects the percentage of total relevant documents retrieved within the retrieved documents. In this case selective random selection retrieved has percentage of 87% whereas the second technique retrieved only 36%.

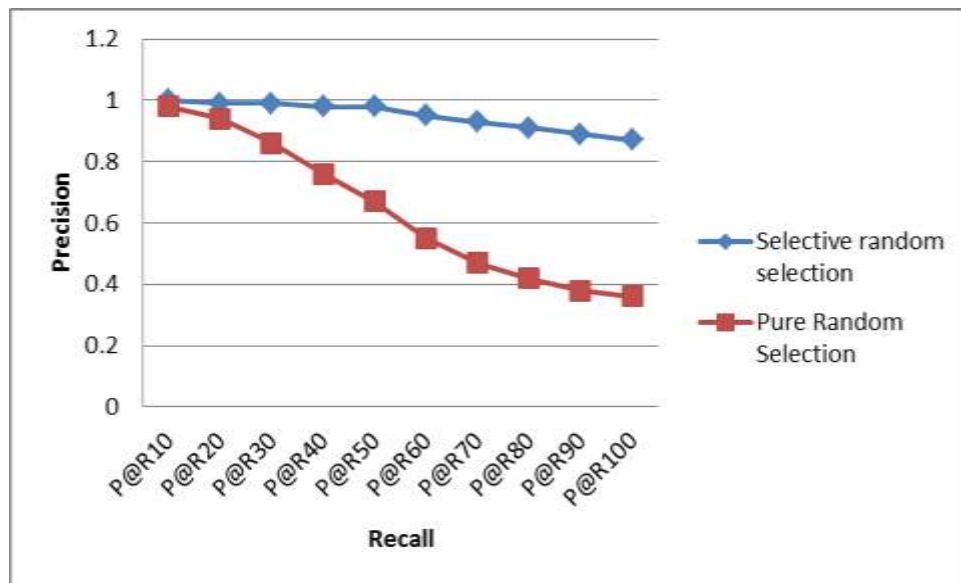


Figure 4-3: Comparison of P@R for different selection techniques.

From this analysis, and the example provided in Appendix D, it is shown that the performance of the selective random selection is better almost 100% than that of the pure random selection in terms of the number of the relevant documents retrieved.

Table 4-8: The P@R enhancement percentage for different selection techniques

Measure	Selective Random Selection	Pure Random Selection	% of improvement
P@R10	1	0.98	2.04
P@R20	0.99	0.94	5.32
P@R30	0.99	0.86	15.12
P@R40	0.98	0.76	28.95
P@R50	0.98	0.67	46.27
P@R60	0.95	0.55	72.73
P@R70	0.93	0.47	97.87
P@R80	0.91	0.42	116.67
P@R90	0.89	0.38	134.21
P@R100	0.87	0.36	141.67
Average	0.95	0.64	66.08

4.7.3 Comparing the selection operators of GA unit in IRUGA

The concept of GA is to produce several generations, until an optimal solution is found in generation G_k where its performance cannot be further improved. Denoting the current generation $G_i = \{p_1, p_2, \dots, p_k\}$, where k is the generation size, then new generation is generated using crossover applied on two selected parents such that the offspring O_i is produced from p_i and p_j . The selection technique of p_i and p_j adopted in the GA unit of IRUGA is the binary tournament selection, in which two random individuals are selected, then the one with higher fitness is nominated as the first parent, and the second parent is selected in a similar way.

However, the binary tournament selection can be applied in several ways. After selecting two candidates for tournament, there are two options among others for nominating the parent. The first option is to pick the fitter one with probability equal to 100%, and this option is called *parent selection-100*. The second option is to select the fitter one with a lower probability and give a chance for the less fit one to participate in the crossover as it may contain some good genes. In this experiment, the comparison is between these two techniques whereas in the second one the probability of selecting the best one is 75%, giving a chance of 25% for the lower candidate to be selected for crossover. Hence, this technique

is called *parent selection-75*. According to this experiment, it is shown that the performance in terms of speed of convergence is almost the same with little improvement in the *parent selection-100*. From the convergence point of view, it is found that *parent selection-100* technique converges after 22.26 generations in average while *parent selection-75* converges after 22.9 generations, which results in a difference of 2.87%. Table 4-5 (which is presented in Section 5.6.1) shows the convergence of these techniques.

Looking at the precision measure (P@N), it is found that the *parent selection-100* technique achieves noticeable enhancement over *parent selection-75* technique. The enhancement ranges from 4.43% at P@10 to 25.57% at P@100 for P@N measure as shown in Figure 4-4 and illustrated numerically in Table 4-9. This reflects the effect of including some low performance chromosomes in the process of crossover. Such chromosomes will introduce low relevance or irrelevant documents into the chromosome; hence they will end with a solution having a lower number of relevant documents.

For the recall measure which is R@N, the results in Figure 4-5 show slight enhancement of *parent selection-100* technique over *parent selection-75* technique. The difference ranges from 1.12% at R@50 to 14.04% at R@10. However, *parent selection-100* was able to retrieve 94% of relevant documents at top100 ranked documents. This is very close to the *parent selection-75* technique which retrieved 92% of such documents. These results are illustrated in Table 4-10.

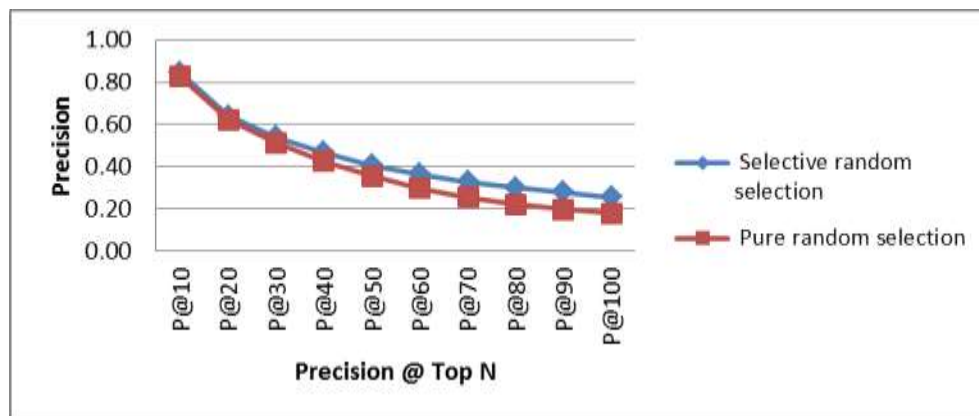


Figure 4-4: Comparison of P@N for different parent selection techniques

Table 4-9: The P@N enhancement percentage for different parent selection techniques

Measure	Parent Selection-100	Parent Selection-75	% of improvement
P@10	0.85	0.83	2.24
P@20	0.64	0.62	2.96
P@30	0.54	0.51	4.96
P@40	0.47	0.42	9.92
P@50	0.41	0.36	14.74
P@60	0.36	0.30	22.73
P@70	0.33	0.25	28.90
P@80	0.30	0.22	34.78
P@90	0.28	0.20	40.96
P@100	0.26	0.18	43.80
Average	0.44	0.39	20.60

Considering the Precision @ Recall measure, it is noted that both *parent selection-75* and *parent selection-100* are very close in performance to each other from P@R10 to P@R60 as illustrated in Figure 4-6. In this range, the performance of *parent selection-100* technique is better than that of the *parent selection-75* technique, whereas the enhancement of the former technique ranges from 0.82% to 3.25%. After that, the difference in performance starts to increase till it reaches 11.01% at P@R100. At this point the P@R100 score for the *parent*

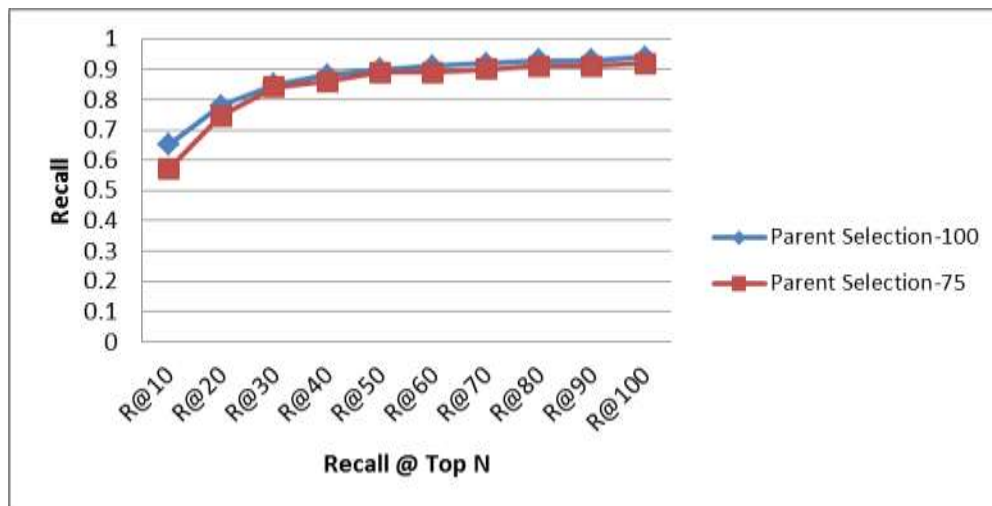


Figure 4-5: Comparison of R@N for different parent selection techniques

Table 4-10: The R@N enhancement percentage for different selection techniques

Measure	Parent Selection-100	Parent Selection-75	% of improvement
R@10	0.65	0.57	14.04
R@20	0.78	0.75	4.56
R@30	0.85	0.84	1.19
R@40	0.88	0.86	2.33
R@50	0.90	0.89	1.12
R@60	0.91	0.89	2.25
R@70	0.92	0.90	2.22
R@80	0.93	0.91	2.20
R@90	0.93	0.91	2.20
R@100	0.94	0.92	2.17
Average	0.87	0.84	3.43

selection-100 technique is 0.87. This means that when retrieving all relevant documents and displaying them to the user, the displayed list will include 13% of irrelevant documents, while the *parent selection-75* technique will include 22% of irrelevant documents in the displayed results. Details are included in Table 4-11.

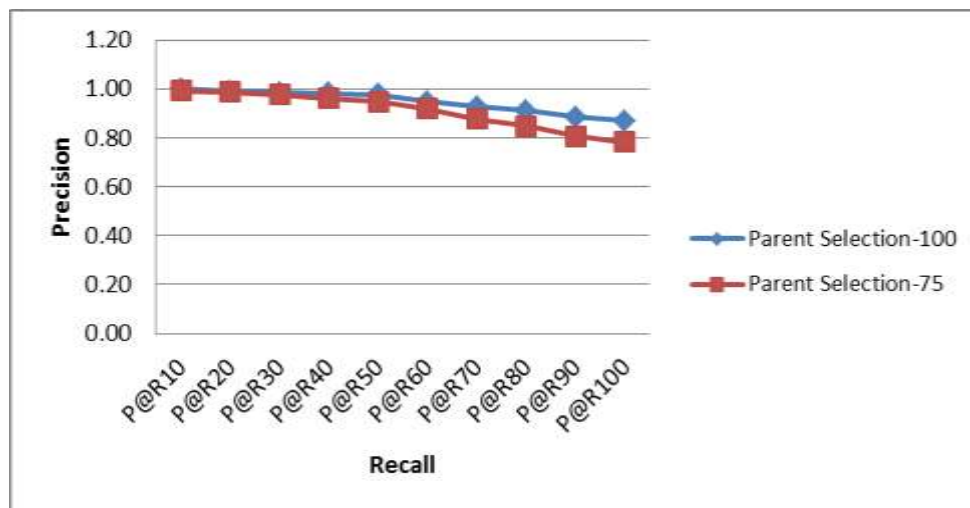


Figure 4-6: Comparison of P@R for Different Parent Selection Techniques

For all measures described above, the *parent selection-100* has a better performance than the *parent selection-75* in terms of speed of convergence, precision at top N , recall at top N and precision at recall. From another point of view, these figures and tables of the results prove

that the assumption stated in Section 3.4.1 is not always true. This assumption states that there are some low performance individuals that may include good genes in the chromosome.

Table 4-11: The P@R enhancement percentage of for Different Parent Selection Techniques

Measure	Parent Selection-100	Parent Selection-75	% of improvement
P@R10	1.00	0.99	0.82
P@R20	0.99	0.99	0.25
P@R30	0.99	0.98	1.37
P@R40	0.98	0.96	1.85
P@R50	0.98	0.95	3.41
P@R60	0.95	0.92	3.25
P@R70	0.93	0.88	6.10
P@R80	0.91	0.85	7.40
P@R90	0.89	0.81	10.52
P@R100	0.87	0.78	11.01
Average	0.95	0.91	4.60

4.7.4 Comparing the Crossover operators of GA unit in IRUGA

In this section, three experiments are performed to study the performance of the *hybrid crossover* technique which performs a one-point crossover on ordered parents to produce one ordered offspring. The first experiment is to compare the hybrid crossover with the two-point crossover, but both will be applied on ordered parents and produce one offspring. The second experiment studies the effect of applying a crossover on ordered and non-ordered parents. However, both techniques will use a one-point crossover and will produce one offspring. The third experiment will examine the effect of producing one offspring and two offspring out of two ordered parents while using a one-point crossover. The results obtained from these three experiments will be analyzed in terms of the speed of convergence and in terms of precision @ N, recall @ N, and precision @ recall.

4.7.4.1 The Comparison between the Hybrid Crossover and Two-Point Crossover

The first experiment in the crossover comparisons is to study the first measure denoted as precision @ top N. In this experiment, the comparison will be done between the hybrid

crossover technique and the two-point crossover, abbreviated as “2-point CO” (Refer to Section 2.4.5.2 for the explanation of the two-point crossover technique).

The first experiment in the crossover comparisons is to apply the GA unit of IRUGA using the classical two-point crossover (Pathak, Gordon and Fan, 2000; Beasley, Bull, and Martin, 1993b; Yang, Korfhage, and Rasmussen, 1992; Spears and De Jong, 1991; Atsumi, 1997). After selecting two parents p_1 and p_2 , two positions are to be selected randomly cp_1 and cp_2 such that $cp_1 < \text{length}(p_1)/2$ and $\text{length}(p_1)/2 < cp_2 < \text{length}(p_1)$. The genes between these two cross positions are exchanged between p_1 and p_2 , knowing that the replaced genes are unique within each chromosome (offspring). Both offspring must have the same length as the parents. When creating offspring O_1 , if a gene g_{li} is found to be already exist in offspring O_1 then it is skipped and no exchange is done, and the genes after this position are shifted to the left. After creating the offspring in this manner, if the offspring has a length smaller than the parents; then the remaining positions are filled by genes after the cp_2 of parent 2 (Refer to Figure 2-4 in Chapter 2 to see an example of the two-point crossover).

Since the genes within each parent are ordered according to the fitness value, it is expected that in the two-point crossover, the offspring will not much differ from the parents, as the genes at each edge forming the extremes of the best and worst documents are copied as they are to the offspring, while the middle genes which have medium performance are exchanged causing the offspring to differ slightly from the parents. This expectation is proved when evaluation measures are analyzed.

Figure 4-7 and Table 4-12 show the precision @ top N retrieved documents. It is shown that the GA unit of IRUGA using *hybrid crossover* has much better performances than the 2-point crossover (referred to as “2-point CO” in this figure) for the reason mentioned above. Moreover, the *hybrid crossover* achieves 0.86 at the top 10 retrieved documents, while the 2-point crossover achieves only 0.34. In other words, *hybrid crossover* achieves an improvement of 152.84% at top 10 over the 2-point crossover.

The second measure to be considered in evaluating this technique is the recall @ top N. Figure 4-8 shows that the recall @ top N retrieved for the *hybrid crossover* starts from 63% until it reaches 85% at R@60. That means this technique is capable of retrieving 85% of the

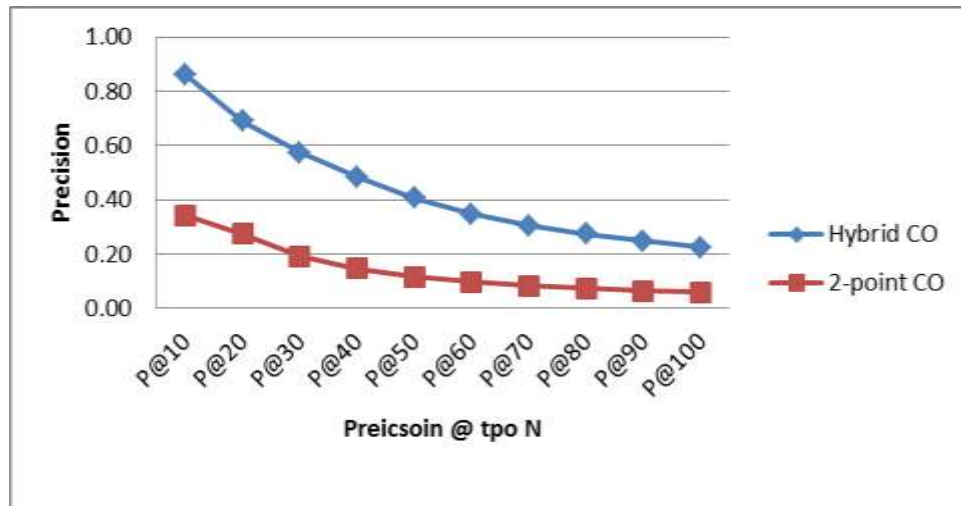


Figure 4-7: Comparison of P@N between hybrid crossover and 2-point crossover technique

total relevant documents at top 60 retrieved documents. However, the 2-point crossover technique starts by retrieving 31% of relevant documents at top 10, and as a whole it retrieves only 35% at top 100 retrieved documents. That implies *hybrid crossover* achieves enhancement of 104% at R@10 and drops to 82.32% at R@100. These results are shown in Table 4-13.

Table 4-12: The P@N enhancement percentage of hybrid crossover over the 2-point crossover techniques

Measure	Hybrid CO	2-point CO	% of improvement
P@10	0.86	0.34	152.84
P@20	0.69	0.27	152.52
P@30	0.57	0.19	199.22
P@40	0.48	0.15	233.22
P@50	0.41	0.12	250.46
P@60	0.35	0.10	259.51
P@70	0.30	0.08	267.68
P@80	0.27	0.07	276.05
P@90	0.25	0.06	283.58
P@100	0.22	0.06	285.64
Average	0.44	0.14	236.07

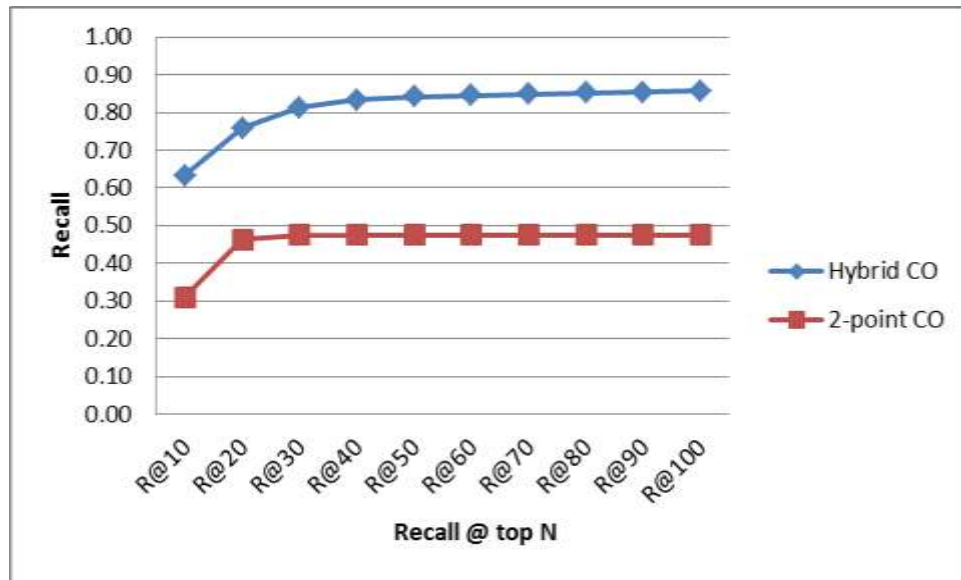


Figure 4-8: Comparison of R@N between hybrid crossover and 2-point crossover technique.

The third measure is the precision @ recall which evaluates the precision percentage when retrieving multiples of 10% of relevant documents. In other words, this measure evaluates the purity of the results from the irrelevant documents.

Table 4-13: The R@N enhancement percentage of hybrid crossover over the 2-point crossover techniques

Measure	Hybrid CO	2-point CO	% of improvement
R@10	0.63	0.31	104.50
R@20	0.76	0.46	63.55
R@30	0.81	0.47	71.09
R@40	0.83	0.48	75.25
R@50	0.84	0.48	77.09
R@60	0.85	0.48	77.72
R@70	0.85	0.48	78.37
R@80	0.85	0.48	79.02
R@90	0.85	0.48	79.67
R@100	0.86	0.48	80.32
Average	0.81	0.46	78.66

Figure 4-9 shows the performance of the proposed *hybrid crossover* over the 2-point crossover. By using the *hybrid crossover*, the GA unit of IRUGA was able to achieve 99% of relevance when retrieving 30% of the total relevant documents. This percentage reduces

to 87% when retrieving all the relevant documents. However, the two-point crossover has 50% of relevant documents when retrieving 30% of relevant documents, and this percentage dropped to 31% when retrieving all relevant documents. These scores show that the *hybrid crossover* managed to achieve an enhancement of 130.07% in the average over all 10-points shown in Table 4-14 for the precision @ recall measure.

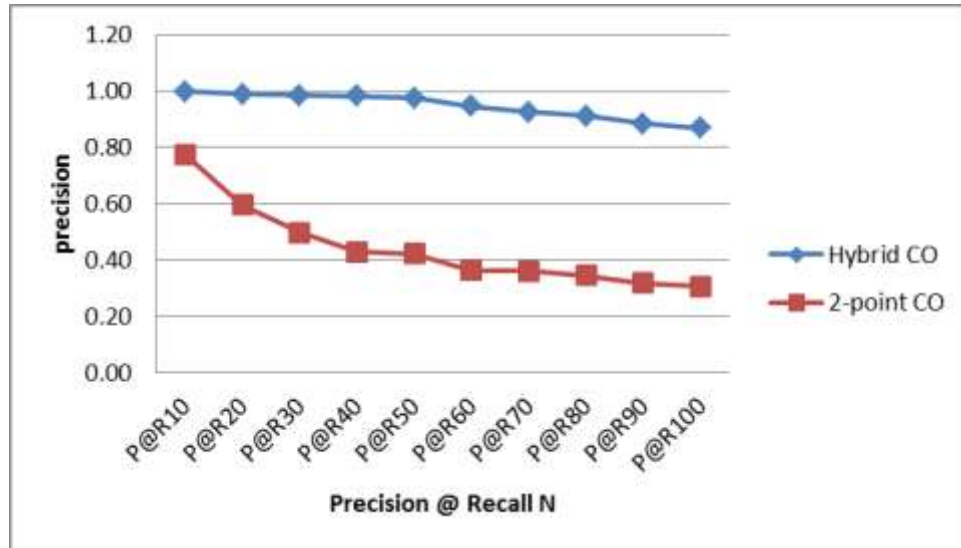


Figure 4-9: Comparison of P@R between hybrid crossover and 2-point crossover technique

Table 4-14: The P@R enhancement percentage of hybrid crossover over the 2-point crossover techniques

Measure	Hybrid CO	2-point CO	% of improvement
P@R10	0.99	0.78	28.96
P@R20	0.99	0.60	66.07
P@R30	0.99	0.50	98.65
P@R40	0.98	0.43	127.68
P@R50	0.98	0.42	131.82
P@R60	0.95	0.36	162.26
P@R70	0.93	0.36	157.96
P@R80	0.91	0.34	164.13
P@R90	0.89	0.32	179.39
P@R100	0.87	0.31	183.77
Average	0.95	0.44	130.07

4.7.4.2 *The Comparison between the Hybrid Crossover and the two-Offspring Crossover*

The second technique of implementing the crossover that needs to be compared with the hybrid crossover is the two-offspring crossover (abbreviated as 2-offspring CO). In this technique, the crossover is applied to two ordered parents to produce two offspring using one crosspoint. Remember that the hybrid crossover produces only one offspring. In this kind of crossover, the genes after the crosspoint are exchanged between the two parents. Hence, the two offspring from parents P_1 and P_2 using crosspoint c_p are expressed as follows:

$$O_1 = \begin{cases} P_{1i}, i \leq c_p \\ P_{2i}, i > c_p \end{cases}$$

$$O_2 = \begin{cases} P_{2i}, i \leq c_p \\ P_{1i}, i > c_p \end{cases}$$

where P_{1i} is the gene i in parent 1, and P_{2i} is the gene i in parent 2.

Looking at the performance of the two-offspring crossover, one can argue that applying *hybrid crossover* is tremendously much better than applying two -offspring crossover. The performance of this technique in terms of P@N is illustrated in Figure 4-10. Once again, the precision @ top N measure of the hybrid crossover is much better than this technique, where the former achieves 0.86 @ 10, while this technique (producing 2-offspring) achieves only 0.48. Although this technique is the second best technique after the hybrid crossover, the hybrid crossover achieved an improvement of 78.49% over this technique as illustrated in Table 4-15.

For R@ top N measure, it is found that the two-offspring crossover technique retrieves 32% of relevant documents at the top 10 and increases to 35% at the top 100 retrieved documents. These results are poor compared with those of the hybrid crossover, which retrieves 63% at the top 10 and 86% at the top 100 retrieved documents. In other words, the hybrid crossover was able to enhance the performance from 98.42% at the top 10 retrieved

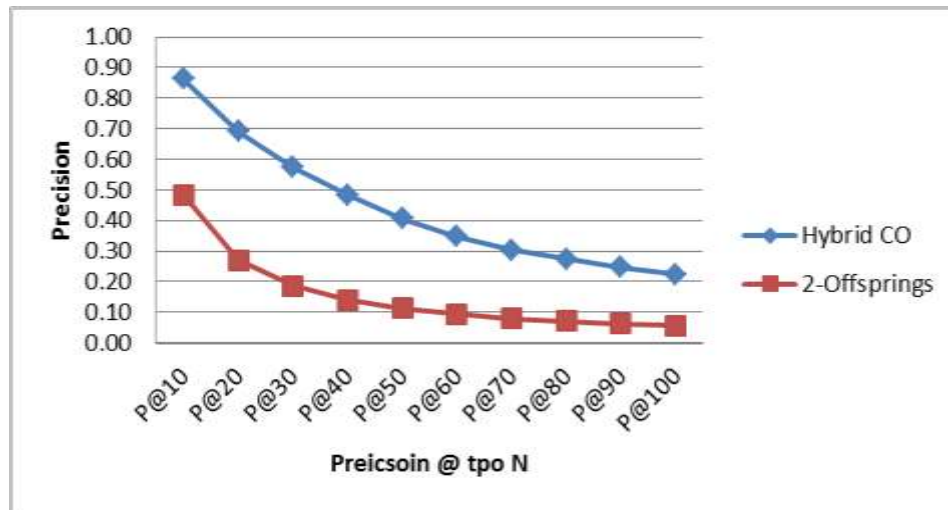


Figure 4-10: Comparison of P@N between hybrid crossover and 2-Offspring crossover techniques.

Table 4-15: The P@N enhancement percentage of hybrid crossover over the 2- Offspring crossover techniques

Measure	Hybrid CO	2- Offspring	% of improvement
P@10	0.86	0.48	78.49
P@20	0.69	0.27	155
P@30	0.57	0.19	207.62
P@40	0.48	0.14	244
P@50	0.41	0.11	261.8
P@60	0.35	0.09	271.18
P@70	0.3	0.08	279.62
P@80	0.27	0.07	288.1
P@90	0.25	0.06	295.82
P@100	0.22	0.06	298.12
Average	0.44	0.16	237.97

documents to 147.12% at the top 100 retrieved documents. The poor performance of this technique is due to the fact that the good genes are not gathered in one chromosome. Therefore each crossover results in splitting the good genes between the two produced offspring. These scores are illustrated graphically in Figure 4-11 and in tabular form in Table 4-16.

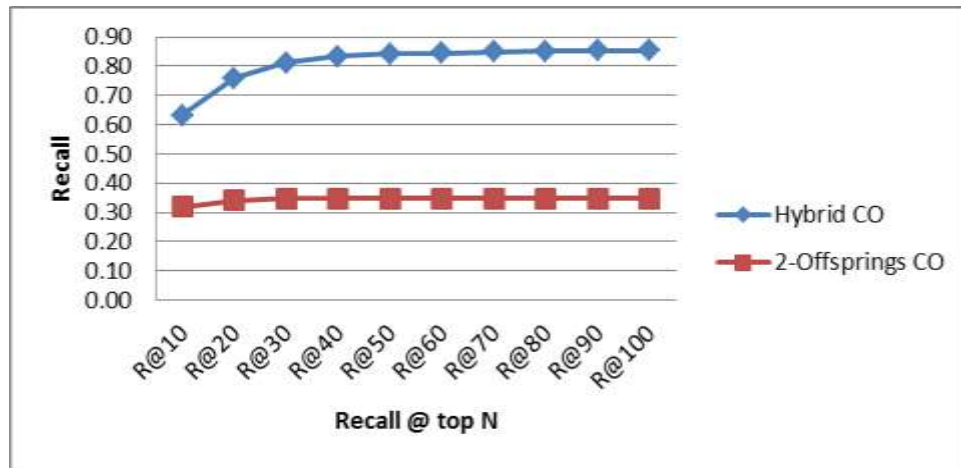


Figure 4-11: Comparison of R@N between hybrid crossover and 2-Offspring crossover techniques.

Table 4-16: The R@N enhancement percentage of hybrid crossover over the 2- Offspring crossover techniques

Measure	Hybrid CO	2- Offspring	% of improvement
R@10	0.63	0.32	98.42
R@20	0.76	0.34	122.37
R@30	0.81	0.35	134.55
R@40	0.83	0.35	140.17
R@50	0.84	0.35	142.69
R@60	0.85	0.35	143.55
R@70	0.85	0.35	144.44
R@80	0.85	0.35	145.34
R@90	0.85	0.35	146.23
R@100	0.86	0.35	147.12
Average	0.81	0.34	136.49

The third measure to be analyzed when comparing the hybrid crossover with the 2-offspring crossover technique is the P@R measure. From Figure 4-12, one can deduce the high difference in performance between the two techniques, where hybrid crossover reaches its maximum precision value of 1 when retrieving 10% (P@R10) of relevant documents, whereas the two-offspring crossover technique reaches its maximum of 0.79 at the same point. This gives an advantage of the proposed hybrid crossover technique which achieves maximum enhancement percentage of 188.78 at P@R100 as illustrated in Table 4-17. On average, the hybrid crossover technique has enhanced the P@R measure by 114.69% on average.

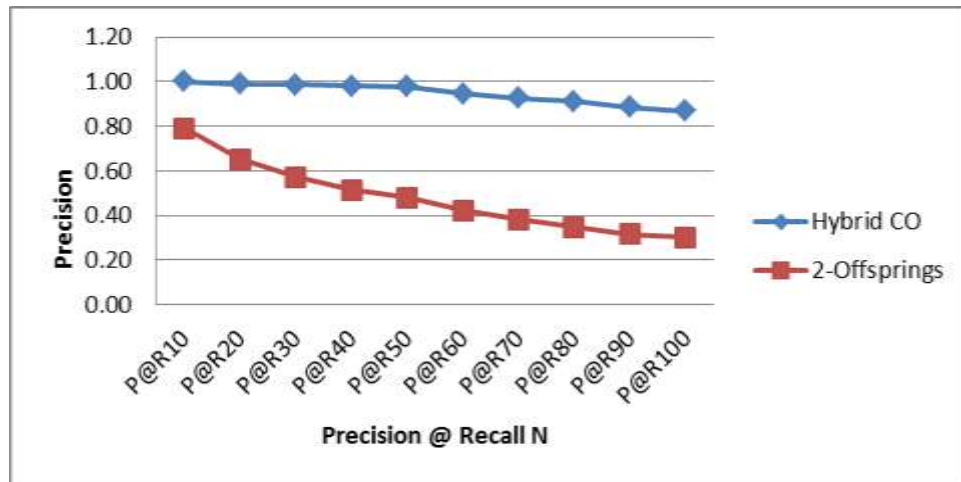


Figure 4-12: Comparison of P@R between hybrid crossover and 2-Offspring crossover techniques..

Table 4-17: The P@R enhancement percentage of hybrid crossover over the 2- Offspring crossover techniques.

Measure	Hybrid CO	2- Offspring	% of improvement
P@R10	1	0.79	26.52
P@R20	0.99	0.65	51.56
P@R30	0.99	0.57	73.37
P@R40	0.98	0.51	90.59
P@R50	0.98	0.48	103.75
P@R60	0.95	0.42	125.08
P@R70	0.93	0.38	143.24
P@R80	0.91	0.35	162.51
P@R90	0.89	0.32	181.5
P@R100	0.87	0.3	188.78
Average	0.95	0.48	114.69

4.7.4.3 Comparing the Hybrid Crossover and Non-ordered Crossover

Another alternative technique for crossover is the one-point crossover applied to non-ordered chromosomes (abbreviated as Non-Ordered CO) to produce one offspring. What differentiates this technique from the hybrid crossover technique is that the genes within the chromosome are not ordered according to their fitness value. Thus, good genes (genes that have high fitness value) are scattered throughout the chromosome resulting in a chromosome having a mixture of good and bad genes distributed arbitrarily within the chromosome. Applying a one-point crossover on such a chromosome results in swapping

these mixed genes from one side of the cross point to the other side without any noticeable improvement.

Although non-ordered crossover techniques is much better than the two-point crossover and two-offspring crossover mentioned earlier, it is still not able to beat the proposed hybrid crossover. Referring to Figure 4-13 of P@N measure, it is shown that this technique starts at a precision of 0.86 at the top 10 retrieved documents and ends with a precision of 0.22 at the top 100 retrieved documents, as compared with 0.58 and 0.13 for the same points for the hybrid crossover technique. That means the second technique is enhanced from 48.12% to 70.62%. These scores are illustrated in Table 4-18.

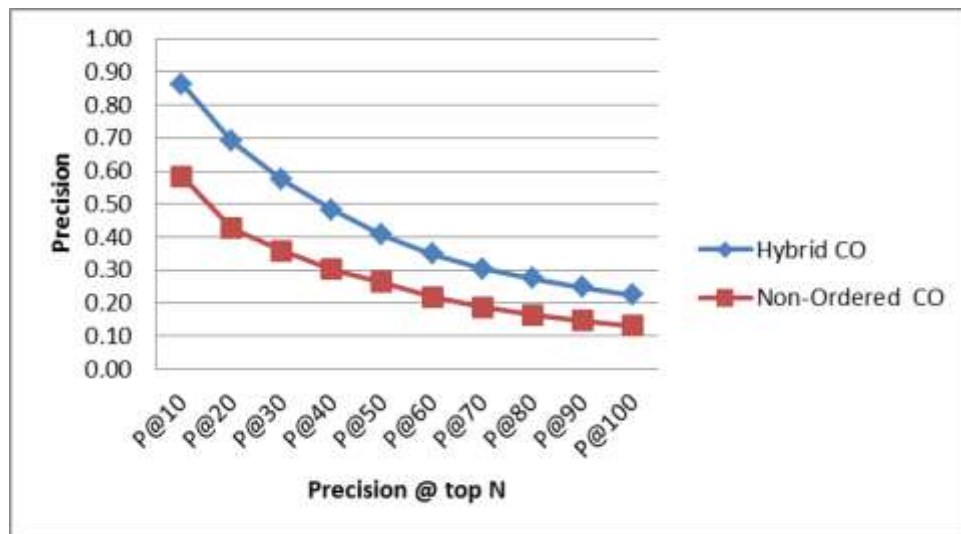


Figure 4-13: Comparison of P@N between hybrid crossover and the non-ordered crossover techniques.

When comparing this technique with the hybrid crossover technique in terms of R@N measure as illustrated in Figure 4-14, it is noticed that the non-ordered crossover

Table 4-18: The P@N enhancement percentage of hybrid crossover over the non-ordered crossover techniques

Measure	Hybrid CO	Non-Ordered CO	% of improvement
P@10	0.86	0.58	48.12
P@20	0.69	0.43	61.95
P@30	0.57	0.36	59.99
P@40	0.48	0.30	59.81
P@50	0.41	0.26	55.05

P@60	0.35	0.22	59.14
P@70	0.3	0.19	62.56
P@80	0.27	0.16	66.79
P@90	0.25	0.15	69.58
P@100	0.22	0.13	70.62
Average	0.44	0.28	61.36

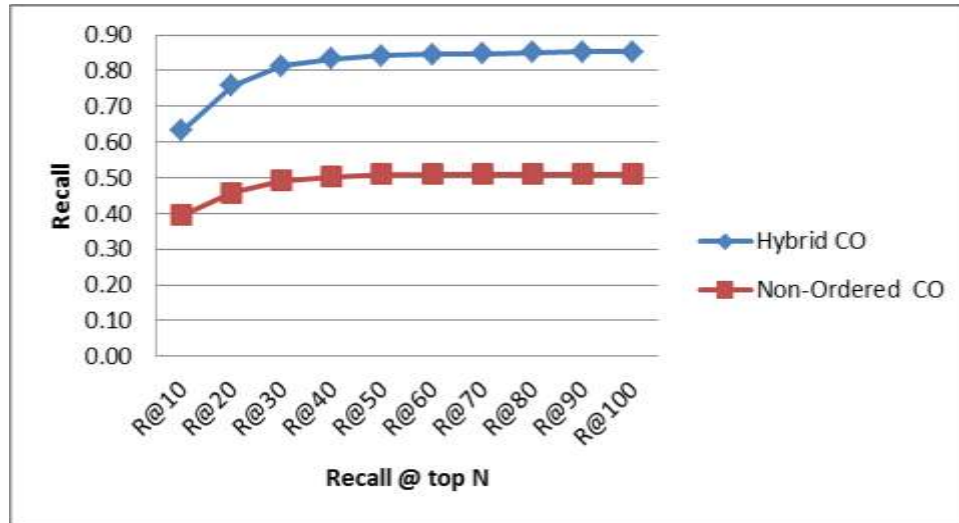


Figure 4-14: Comparison of R@N between hybrid crossover and the non-ordered crossover techniques

Table 4-19: The R@N enhancement percentage of hybrid crossover over the non-ordered crossover techniques.

Measure	Hybrid CO	Non-Ordered CO	% of improvement
R@10	0.63	0.39	60.33
R@20	0.76	0.46	65.50
R@30	0.81	0.49	65.10
R@40	0.83	0.50	65.47
R@50	0.84	0.51	65.09
R@60	0.85	0.51	65.68
R@70	0.85	0.51	66.29
R@80	0.85	0.51	66.90
R@90	0.85	0.51	67.50
R@100	0.86	0.51	68.11
Average	0.81	0.49	65.60

performance ranges between 39% at R@ top 10 and 51% at R@ top 100. This means that this technique lags behind hybrid crossover technique by 60.33% to 68.11%. Table 4-19 lists the scores for each point of the scale of R@N measure.

The last measure to be compared between the non-ordered crossover technique and the hybrid crossover is the precision @ recall measure. The results are shown in Figure 4-15. The performance of the former technique ranges between 0.92 at P@R10 and 0.43 at P@R100, compared with hybrid crossover which ranges from 1 at P@R10 to 0.87% at P@R100. As demonstrated in Table 4-20, it is found that the proposed technique enhanced the performance by 102.89% at P@R100.

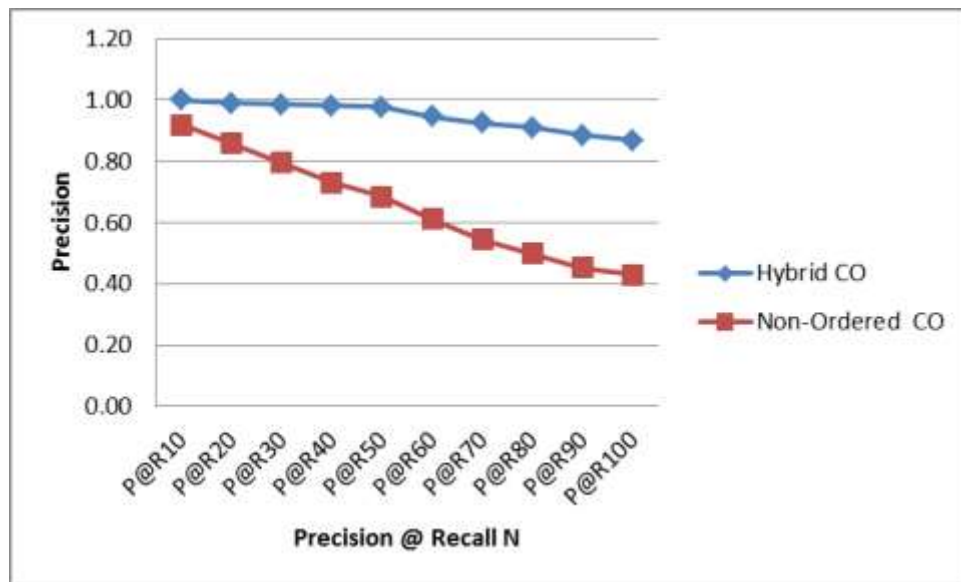


Figure 4-15: Comparison of P@R between hybrid crossover and the non-ordered crossover techniques

4.7.4.4 Revisiting the Crossover Operator of the GA unit of IRUGA

In the above analysis of the precision score for the hybrid crossover, it is found that it achieved 0.86 at P@10 and 0.69 at P@20. Since the genes are ordered within the chromosome, it is expected to have a higher percentage than 86% for precision @ 10 where the best genes of each candidate chromosome are pushed to the new offspring. Remember that P@10 means the number of relevant documents at the top 10 retrieved documents.

Table 4-20: The P@R enhancement percentage of hybrid crossover over the non-ordered crossover techniques.

Measure	Hybrid CO	Non-Ordered CO	% of improvement
P@R10	1	0.92	8.69
P@R20	0.99	0.86	15.29
P@R30	0.99	0.8	24.5
P@R40	0.98	0.73	34.01
P@R50	0.98	0.68	43.32
P@R60	0.95	0.61	55.35
P@R70	0.93	0.54	70.73
P@R80	0.91	0.5	82.58
P@R90	0.89	0.45	96.97
P@R100	0.87	0.43	102.89
Average	0.95	0.65	53.43

However, what causes this percentage to drop is the high percentage of queries that have relevant documents less than 10. In the previous experiment these queries form 47% of the total queries. In such a case, exactly 53% of the queries will have a precision less than one. Moreover, this score will depend also on the number of relevant documents per query. So if the relevant number of documents for these queries is low, the score will also be low.

For example, if 53% of these documents have 5 relevant documents, then the precision @10 for these documents in the best case, i.e. if all relevant documents are retrieved, will be 5/10 which is 0.5.

In other words, this measure depends on two factors. The first one is the number of queries that have a high number of relevant documents, and the second factor is the number of relevant documents per query.

Consequently, this experiment is repeated by using only the queries that have 10 relevant documents or more. The result of this experiment is shown in Figure 4-16, where the precision@10 reaches its maximum of 1. This score is 14% better than the score of the previous experiment. Moreover, this score reflects the high effect of the previously mentioned factors on the P@N measure.

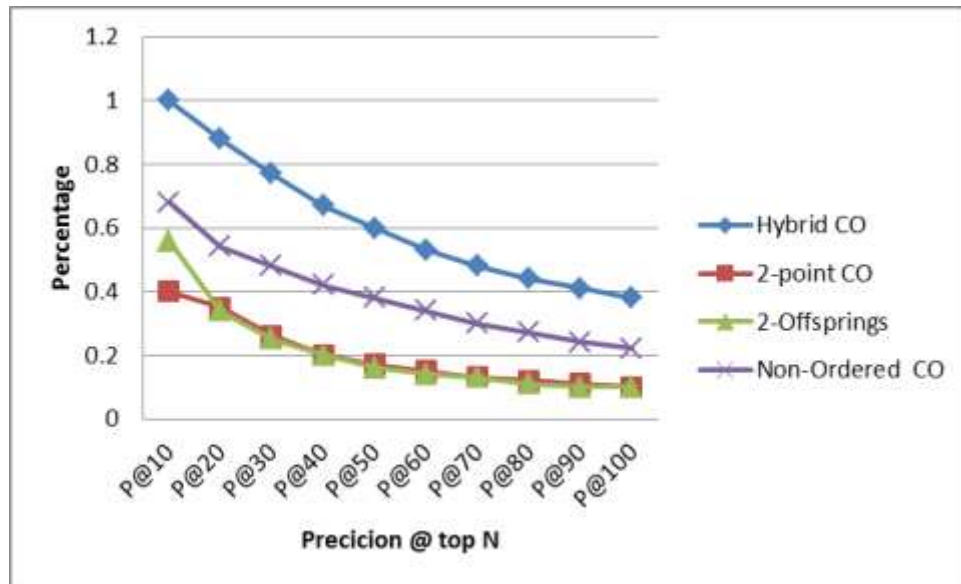


Figure 4-16: Comparison of P@N for Different Crossover Techniques for queries having more than 10 relevant documents.

From another point of view this result shows the tremendous performance of the hybrid crossover over the other crossover techniques. The score of the second best crossover technique, which is the non-ordered crossover, is 0.68 while the scores of two-point crossover and two-offspring crossover are only 0.4 and 0.56 respectively. The percentages of enhancement achieved by the hybrid crossover over other crossover techniques are represented in Table 4-21.

Generally speaking, the proposed hybrid crossover technique performance is much better than other crossover techniques analyzed in this study. The advantage of this technique comes from the mechanism it follows in selecting the best genes from the chromosome and gives it a better chance to be inherited to the next generation.

Moreover, the GA unit of IRUGA shows the strength of the hybrid crossover technique when it is applied to queries that have a high number of relevant documents, especially when comparing the results for the top retrieved documents where $N \leq 20$.

Table 4-21: The P@N improvement of hybrid crossover over other crossover techniques for queries having more than 10 relevant documents.

Measure	Hybrid CO	2-point CO	% of improvement	2-Offspring	% of improvement	Non-Ordered CO	% of improvement
P@10	1.00	0.40	152.84	0.56	78.49	0.68	48.12
P@20	0.88	0.35	152.52	0.34	155.00	0.54	61.95
P@30	0.77	0.26	199.22	0.25	207.62	0.48	59.99
P@40	0.67	0.20	233.22	0.20	244.00	0.42	59.81
P@50	0.60	0.17	250.46	0.16	261.80	0.38	55.05
P@60	0.53	0.15	259.51	0.14	271.18	0.34	59.14
P@70	0.48	0.13	267.68	0.13	279.62	0.30	62.56
P@80	0.44	0.12	276.05	0.11	288.10	0.27	66.79
P@90	0.41	0.11	283.58	0.10	295.82	0.24	69.58
P@100	0.38	0.10	285.64	0.10	298.12	0.22	70.62
Average	0.62	0.20	236.07	0.21	237.98	0.39	61.36

4.7.5 Testing Different Fitness Functions

For our ranking technique, the decision about whether to take or reject a document depends only on the value computed by the proposed fitness function. The proposed fitness function is developed based on local factors only to make the evaluation of the document independent of other documents. The local factors are those obtained from the document under consideration such as document size, number of unique terms within the document, and the total number of specific terms within the document.

As mentioned in Section 4.4.2, the performance of term-proximity fitness function will be examined against two well known fitness functions in the IR domain. These fitness functions are the Okapi-BM25 and the Bayesian inference network model functions. These functions are listed in Table 4-22.

Table 4-22: List of fitness functions

Fitness method	Fitness Formula
Okapi-BM25	$f(D) = \sum_{T \in Q} \frac{(k1 + 1) \times tf}{(k1 \times ((1 - b) + b \frac{length}{length_{avg}}) + tf)} \times \log \frac{N - df + 0.5}{df + 0.5}$
Bayesian inference network model	$w_i = (d_t \cdot H + (1 - d_t)) \cdot \frac{\log(tf_i + 0.5)}{\log(\max tf_i + 1.0)} \cdot \frac{\log(\frac{N}{n})}{\log N}$

Term-proximity fitness function	$f(D) = a \frac{\sum_{i=1}^K k_{ui}}{K} + b \frac{\sum_{i=1}^K k_{ui} - 1}{\sum_{i=1}^{K-1} \min(d_{i,i+1})} + c \frac{1}{\text{avg}(\sum_{i=1}^K \min(p_i))} + d \log\left(\frac{\sum_{i=1}^K w_i}{\sum_{i=1}^K k_i}\right)$
---------------------------------	---

It is obvious from the results shown in Figure 4-17 that the GA unit of IRUGA which uses the term-proximity function has the highest average precision of 86% in the first top 10 ranked documents at the moment where the other two models reach only 49% for the Bayesian network inference model and 55% for the Okapi-BM25, which means that the proposed system achieves a 75.27% improvement on average in precision at the top 10 ranked documents over the Bayesian model and 31.78% over the OKAPI-BM25 models. Details of these results are illustrated in Table 4-23.

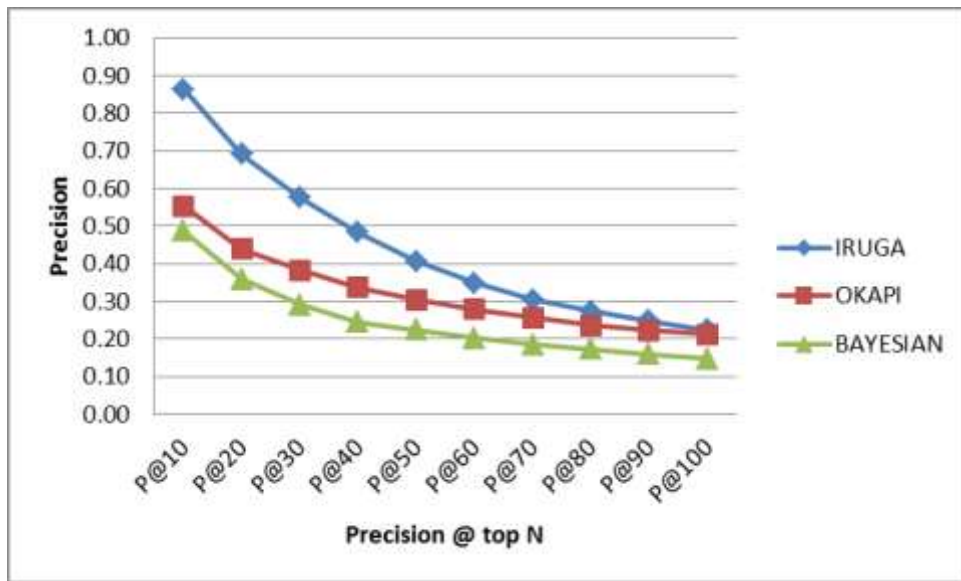


Figure 4-17: Comparison of P@N for Different Fitness Functions

Another measure to be considered here is the recall @ top N measure. The term proximity function was able to retrieve 84% of related documents at maximum of the top 50 retrieved documents, as shown in Figure 4-18, whereas the Bayesian network inference model and the Okapi-BM25 reach only 75% and 71% recall respectively for the first 50 retrieved documents. The improvement of the term proximity model is 22.52% over the Bayesian network inference model and 27.55% over the Okapi-BM25 model. Table 4-24 illustrates the details of these results.

Table 4-23: The P@N enhancement percentage of the term proximity fitness function over other fitness functions

Measure	Term proximity	BAYESIAN	% of improvement	OKAPI	% of improvement
P@0	1.00	0.82	21.95	0.86	18.60
P@10	0.85	0.49	77.73	0.55	56.18
P@20	0.64	0.36	92.66	0.44	57.66
P@30	0.54	0.29	97.62	0.38	49.89
P@40	0.47	0.24	97.79	0.34	44.03
P@50	0.41	0.22	81.95	0.30	33.73
P@60	0.36	0.20	73.06	0.28	24.73
P@70	0.33	0.18	64.94	0.26	18.50
P@80	0.30	0.17	59.37	0.24	15.94
P@90	0.28	0.16	56.19	0.22	11.64
P@100	0.26	0.15	51.42	0.21	5.51
Average	0.49	0.30	70.43	0.37	30.58

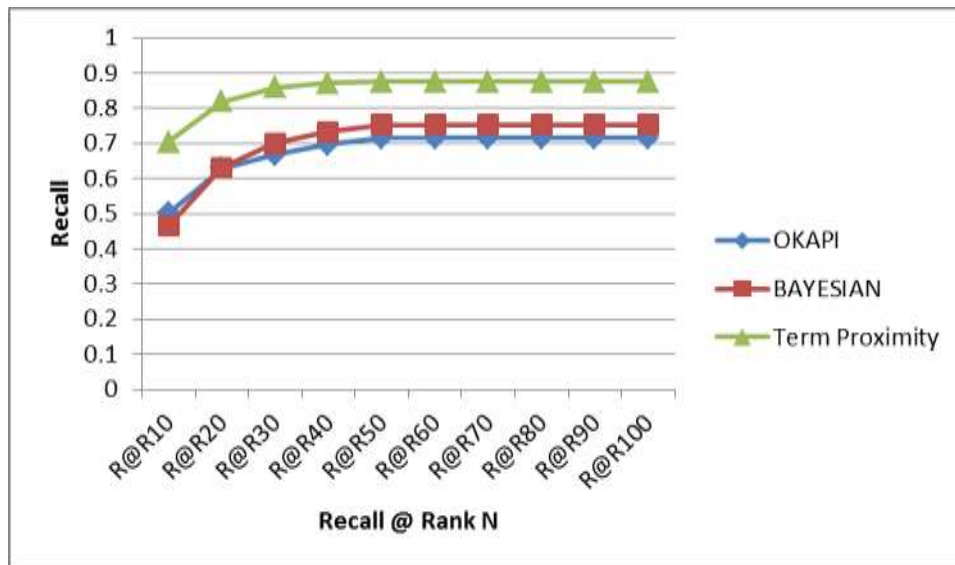


Figure 4-18: Comparison of R@N for Different Fitness Functions

When examining the precision @ recall measure, which is shown in Figure 4-19, one can notice the high performance of the proximity function. The precision starts by 1 when the system retrieves 10% of relevant documents and then reduces gradually until it reaches 0.87 when retrieving all relevant documents. This means that until it retrieves 10% of relevant documents, all the displayed documents are relevant. In fact, this score was not achieved by

Table 4-24: The R@N enhancement percentage of the term proximity fitness function over other fitness functions

Measure	Term proximity	BAYESIAN	% of improvement	OKAPI	% of improvement
R@10	0.65	0.46	40.05	0.50	29.86
R@20	0.78	0.63	23.73	0.63	24.14
R@30	0.85	0.70	21.25	0.67	27.42
R@40	0.88	0.73	20.04	0.70	26.40
R@50	0.90	0.75	19.60	0.71	26.04
R@60	0.91	0.75	20.93	0.71	27.44
R@70	0.92	0.75	22.26	0.71	28.84
R@80	0.93	0.75	23.58	0.71	30.24
R@90	0.93	0.75	23.58	0.71	30.24
R@100	0.94	0.85	10.22	0.75	24.91
Average	0.87	0.71	22.52	0.68	27.55

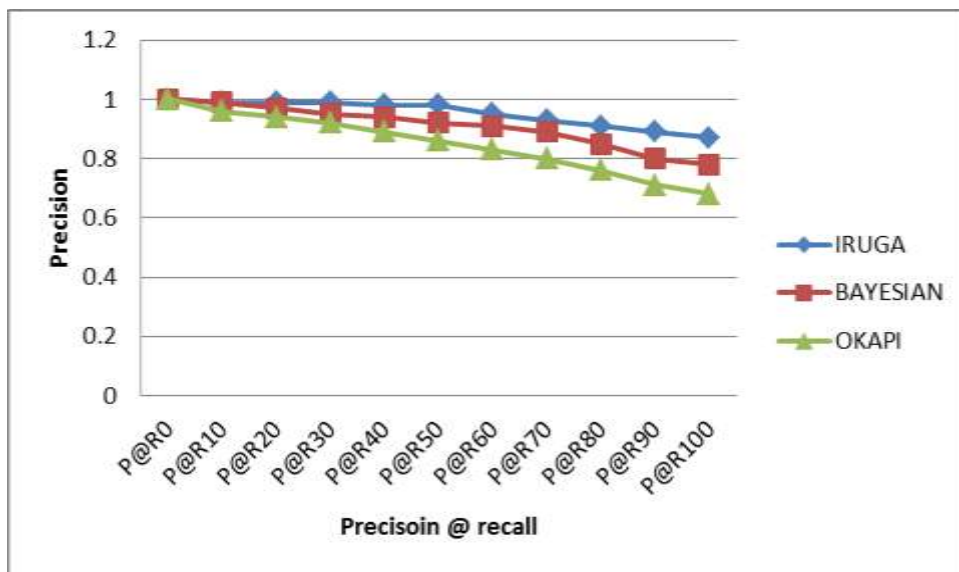


Figure 4-19: Comparison of P@R for Different Fitness Functions

any other technique or model. Moreover, the 0.87 at 100% recall implies that when the system retrieves all the relevant documents, only 13% of those retrieved are not relevant to the user query and they appear in low rank or at the bottom. This result is very close to the user anticipation since he or she is looking to have all top ranked documents as relevant, and most of the relevant documents appear in top position.

These results imply that the term proximity model achieved a 5.16% enhancement compared with the Bayesian inference network model, and achieved an enhancement of 13.17% when compared with the OKAPI-BM25 model. Details of these figures are illustrated in Table 4-25.

Table 4-25: The P@R enhancement percentage of the term proximity fitness function over other fitness functions.

Measure	IRUGA	BAYESIAN	% of improvement	OKAPI	% of improvement
P@R0	1.00	1.00	0.00	1.00	0.00
P@R10	0.99	0.99	1.01	0.96	4.17
P@R20	0.99	0.97	2.06	0.94	5.32
P@R30	0.99	0.95	4.21	0.92	7.61
P@R40	0.98	0.94	4.26	0.89	10.11
P@R50	0.98	0.92	6.52	0.86	13.95
P@R60	0.95	0.91	4.40	0.83	14.46
P@R70	0.93	0.89	4.49	0.80	16.25
P@R80	0.91	0.85	7.06	0.76	19.74
P@R90	0.89	0.80	11.25	0.71	25.35
P@R100	0.87	0.78	11.54	0.68	27.94
Average	0.95	0.91	5.16	0.85	13.17

The reason behind high results for the proposed fitness function is that it doesn't depend only on the frequency of terms within the document as other fitness functions do. It also depends on the importance of the term based on the HTML tag and on the position of the terms within the document, in addition to considering the distance between the terms. At the same time it doesn't ignore the term frequency factor.

4.7.6 Mutation

The last operator to be checked in the GA unit of IRUGA is mutation. This section will examine the performance of the GA unit of IRUGA when implemented using different probabilities of applying mutation. Normally, mutation is applied with low probability to simulate the natural behaviour of an organism. However, because it has many advantages (please refer to Section 2.3.5 which explains them) that enhances the GA unit of IRUGA performance, the mutation rate to be applied in the GA unit of IRUGA is 0.7, which is the maximum rate used (Radwan et al 2006). Hence the comparison in this experiment is done when IRUGA is using mutation with probabilities of 0.001, 0.1, 0.2 and 0.7.

As a general observation, the performance of the GA unit of IRUGA when applying these mutation rates is almost the same and can hardly be distinguished using the graphs. Therefore tables are included here to better represent the figures. Looking at precision @ top N measure, it is found that MUTE70 achieves the highest performance when compared with other rates of mutation. It achieves the highest enhancement when compared with

MUTE001 although it is a slight enhancement. This enhancement ranges from 1.52% at P@10 to 9.67% at P@70. The performance of different mutation rates becomes closer to MUTE70 as the mutation rate increases. Illustration of this is shown in Figure 4-20 where the performance of MUTE20 is the same as that of MUTE70 except for one point which is at P@70. The reason behind this similarity is that in mutation only one gene per chromosome is nominated for replacement. Moreover, the replacement is done only if the new gene has better fitness value than the replaced one. Otherwise, no replacement is done. This is to maintain the chromosome performance at its maximum value.

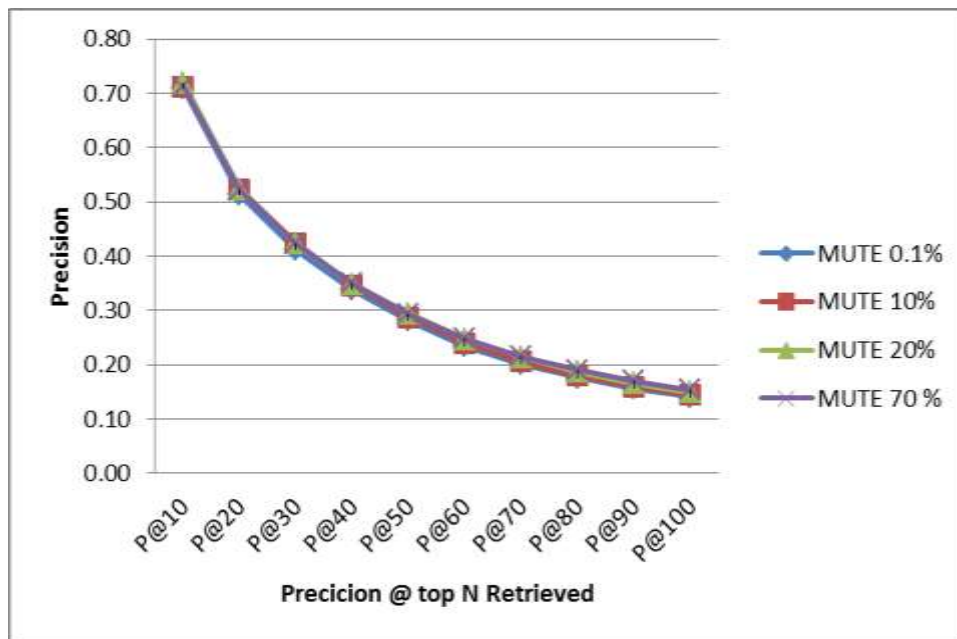


Figure 4-20: Comparison of P@N for Different Mutation Rates

Another measure to be examined is the recall @ top N (Figure 4-21). In contrast to the previous measure, the performance of MUTE70 has a different effect on recall @ top N measure. It can be seen from Table 4-27 that the maximum enhancement of MUTE70 is when compared with MUTE10. Here MUTE70 retrieves 65% of relevant documents at top 10 retrieved documents whereas MUTE10 retrieves 62%, achieving an enhancement of 5.68%. The maximum enhancement occurs at recall @100 where MUTE70 retrieves 94% while MUTE10 retrieves 89% only, resulting in an enhancement of 6.03%.

Table 4-26: The P@N enhancement percentage of MUTE70 over other mutation rates

Mutation type	MUTE70	MUTE001	% of improvement	MUTE10	% of improvement	MUTE20	% of improvement
P@10	0.72	0.71	1.52	0.71	1.31	0.72	0.00
P@20	0.52	0.51	1.33	0.52	-0.64	0.52	0.00
P@30	0.42	0.41	2.21	0.42	-0.84	0.42	0.00
P@40	0.35	0.34	3.60	0.35	1.25	0.35	0.00
P@50	0.29	0.28	3.28	0.29	0.88	0.29	0.00
P@60	0.25	0.28	-10.97	0.24	4.17	0.25	0.00
P@70	0.22	0.20	9.67	0.21	7.13	0.21	4.76
P@80	0.19	0.18	8.12	0.18	5.58	0.19	0.00
P@90	0.17	0.16	9.04	0.16	6.48	0.17	0.00
P@100	0.15	0.14	6.84	0.14	7.14	0.15	0.00
Average	0.33	0.32	3.47	0.32	3.25	0.33	0.48

Details of the performance of all mutation rates and the percentage of enhancement of MUTE70 over others are represented numerically in Table 4-27.

The last measure to be considered in this section is the precision @ recall measure. Figure 4-22 demonstrates the behaviour of MUTE70 as compared with MUTE001, MUTE10 and MUTE20 graphically. At Precision @ recall 10 the MUTE70 once again has the best performance where the precision at 10% recall is 100%. That means when retrieving 10% of relevant documents, all the displayed results are relevant.

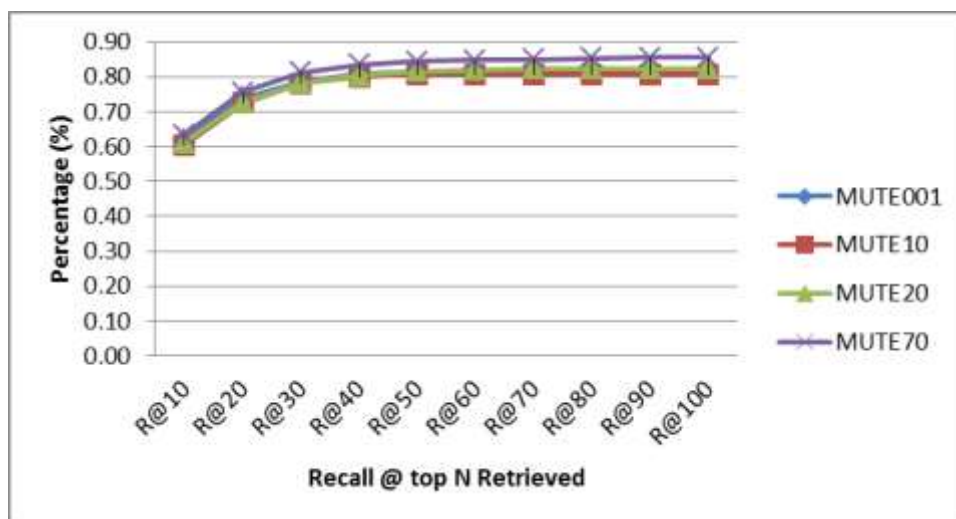


Figure 4-21: Comparison of R@N for different mutation rates.

Table 4-27: The R@N enhancement percentage of MUTE70 over other mutation rates

Mutation type	MUTE70	MUTE001	% of improvement	MUTE10	% of improvement	MUTE20	% of improvement
R@10	0.65	0.64	2.34	0.62	5.68	0.63	3.69
R@20	0.78	0.76	2.17	0.75	4.16	0.75	3.87
R@30	0.85	0.82	3.21	0.82	4.00	0.82	3.80
R@40	0.88	0.85	3.20	0.84	4.32	0.85	3.79
R@50	0.9	0.87	3.52	0.86	4.61	0.87	3.43
R@60	0.91	0.88	3.91	0.87	5.01	0.88	3.82
R@70	0.92	0.88	4.31	0.87	5.41	0.88	4.22
R@80	0.93	0.89	4.67	0.88	5.77	0.89	4.58
R@90	0.93	0.89	4.93	0.88	6.03	0.89	4.83
R@100	0.94	0.90	4.93	0.89	6.03	0.90	4.83
Average	0.87	0.84	3.72	0.83	5.10	0.84	4.08

At precision @ recall 100, the score is very close to that of other mutation rates, where both MUTE70 and MUTE20 achieved 0.87, MUTE001 achieved 0.869, and MUTE10 achieved 0.881. It is noted from Table 4-28 that, at some points, the performance of MUTE001, MUTE10 and MUTE20 is better than that of MUTE70. Although the overall average performance of MUTE10 is better than MUTE70 by 0.5%, this improvement is considered to be very small which gives a favour for the MUTE70 as it achieved the highest precision at top N and the highest recall at top N.

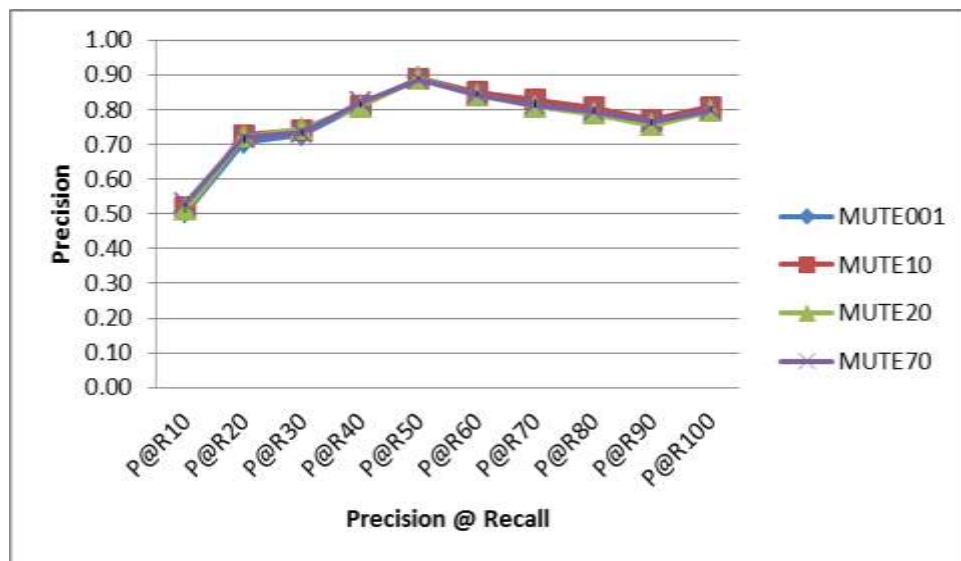


Figure 4-22: Comparison of P@R for different mutation rates

Table 4-28: The P@R enhancement percentage of MUTE70 over other mutation rates.

Mutation type	MUTE70	MUTE001	% of improvement	MUTE10	% of improvement	MUTE20	% of improvement
P@R10	1.000	0.947	5.60	0.977	2.34	0.960	4.17
P@R20	0.990	0.970	2.09	1.000	-0.99	0.992	-0.16
P@R30	0.990	0.978	1.20	0.997	-0.73	1.011	-2.05
P@R40	0.980	0.968	1.21	0.967	1.39	0.967	1.39
P@R50	0.980	0.983	-0.26	0.982	-0.25	0.982	-0.25
P@R60	0.950	0.957	-0.73	0.958	-0.88	0.947	0.30
P@R70	0.930	0.945	-1.57	0.950	-2.11	0.927	0.31
P@R80	0.910	0.914	-0.49	0.921	-1.23	0.899	1.27
P@R90	0.890	0.897	-0.74	0.902	-1.30	0.878	1.33
P@R100	0.870	0.869	0.08	0.881	-1.23	0.870	0.00
Average	0.949	0.943	0.64	0.954	-0.50	0.943	0.63

4.8 Summary

Several experiments are conducted in this chapter to examine the performance of multiple techniques applied to test the performance of IRUGA.

This chapter started by describing the HTML documents set used to test the performance of IRUGA.

Next in the chapter was the identification of the measures that need to be used to evaluate the IRUGA performance. These measures are precision @ rank N, recall @ rank N, and precision @ recall N, where $N \leq 100$ and N is a multiple of 10. In addition to these measures, the speed of convergence was considered as a mean to evaluate different techniques, where the maximum number of generations is obtained for each technique. The smaller the number of generations implies the faster convergence.

As mentioned earlier, IRUGA consists of two main units. The first one is the document index. The proposed indexing model is the enhanced inverted index model. The experiment applied here shows the superiority of this model over other models in terms of space required to store the indexed terms along with associated data where it used 0.5% of the space required to store the same index using the vector space model.

The second unit of IRUGA is the GA. GA is applied by performing several operators to

produce multiple generations before converging and presenting the results at the last generation.

A set of experiments is conducted to compare the performance of the proposed technique of each operator with the main known techniques.

The first experiment tested the *selective random selection* technique of the first generation with *random selection* technique. The enhancement achieved for the former technique on average using this technique was 101.94% for P@N measure, 110.02 for R@N measure, and 66.08% for P@R.

The second experiment was to test the effect of the parent selection technique on the quality of retrieved documents. The comparison was done between *parent selection-100* and *parent selection-75*. The results obtained showed the advantage of the former one for the measures P@N, R@N and P@R with an enhancements of 20.6%, 13.43% and 4.6% respectively.

The third experiment conducted was on crossover techniques. The proposed *hybrid crossover* is tested against: simple 1-point crossover producing two offspring, non-ordered crossover producing one offspring, and 2-point crossover producing one offspring. The *hybrid crossover* technique achieved the best enhancement for all measures over all other techniques, and the maximum enhancement was 237.97% and 136.49 % for P@N and R@N when compared to a simple 1-point crossover producing two offspring. Moreover, the *hybrid crossover* achieves the highest enhancement of 130.07% over the two-point crossover.

The fourth experiment performed in this chapter to compare the performance of the proposed TPDFF function with two of the well know fitness functions used in this domain which are Okapi-BM25 and Bayesian inference network model. The results reflected the superiority of TPDFF over these functions. The improvement achieved by TPDFF over Okapi-BM25 and Bayesian inference network model is 30.58% and 30% respectively in terms of P@N measure, 27.55% and 22.52% for R@N measure, while it is 13.17% and 5.16% in terms of P@R measure.

The last experiment conducted was to test the performance of IRUGA against different mutation rate. The adopted mutation rate for IRUGA was 70% which was compared with

the rates: 0.1%, 10% and 20%. The results obtained show that the mutation rate plays a very minor factor since the average improvement in the best case was 5.1% for R@N when the mutation rate is 10%.

The summary of all experiments including operator, techniques used, results of each measure for each technique and the improvement percentage achieved by IRUGA are included in Table 4-29.

However, still full marks are not achieved for this technique and this is due to the nature of IRUGA which is based on the probabilistic feature of its backbone which is the GA unit.

Table 4-29: Summary of the techniques, measures and percentage of improvement. The percentage of improvement beside each technique represents the average comparison of this technique with the one in bold above it.

Operator	Technique	P@N	Impr.	R@N	Impr.	P@R	Impr.	Convergence speed	Impr.
Initial generation creation	Selective random selection	0.49		0.87		0.95		22.26	
	Pure Random Selection	0.31	101.94	0.44	110.02	0.64	37.80	24.96	12.13
Parent Selection	Parent Selection-100	0.49		0.87		0.95		22.26	
	Parent Selection-75	0.39	20.60	0.86	3.43	0.91	4.6	22.9	2.88
Crossover	Hybrid crossover	0.49		0.87		0.95		22.26	
	2-point crossover	0.14	236.07	0.46	78.66	0.44	130.07	28	25.79
	2-Offspring	0.16	237.97	0.34	136.49	0.48	114.69	43.4	94.97
	Non-Ordered crossover	0.28	61.36	0.51	65.60	0.66	53.43	13.65	-38.68
Mutation	MUTE70	0.33		0.87		0.949		22.26	
	MUTE001	0.32	3.47	0.84	3.72	0.943	0.64	22.13	-0.58
	MUTE10	0.32	3.25	0.83	5.10	0.954	-0.50	22.56	1.35
	MUTE20	0.33	0.48	0.84	4.08	0.943	0.63	22.3	0.18
Fitness function	Proximity term fitness function	0.49		0.87		0.95		21.43	
	Okapi-BM25	0.37	30.58	0.68	27.55	0.83	13.17	41.50	48.36
	Bayesian Network Interface Model	0.30	70.43	0.70	22.52	0.91	5.16	20.84	-2.84

Chapter Five: TPBTIR and IRUGA

5.1 Introduction

IR problem are investigated using many technique as illustrated in Chapter 2. In previous chapters the IR performance was enhanced using a GA-based IR model which is called IRUGA. However, GA in general is a probabilistic technique based on randomness, which due to its nature may miss some high quality documents during the selection of individuals for the initial generation. Therefore, in this chapter, a traditional IR (TIR) model is to be designed in order to overcome the random nature of GA. The proposed TIR is featured by examining all the documents in the collection. Thus, no document is missed during the evaluation process, resulting in providing the highest performance document in the space as response to the user query. In contrast, IRUGA is a probabilistic model where a random sample of the collection is selected and evaluated. Nevertheless, what controls the performance in TIR is the collection size. If the collection size is huge enough, this approach becomes extensive in both time and computation cost giving the advantage for IRUGA.

In order to recognize the novelty of the proposed Term Proximity Evaluation Function (TPEF) which is referred to as TPEF in Section 3.4.4, a comparison between this function and other fitness functions needs to be performed. An empirical study is conducted in this chapter to compare the performance of both IRUGA and TIR using the TPEF evaluation functions. In addition, a comparison will be conducted to compare the performance of TPEF with the performance of the two well known evaluation functions applied in IR and already described in Chapter 4, namely, the OKAPI-BM25 and Bayesian inference network functions. This experiment will be conducted using both IRUGA and TIR models. Moreover, this chapter analyzes the time performance of both IRUGA and TIR when using the TPEF function. With the results analyzed, all these techniques are discussed

At the end of this chapter, the results obtained are discussed by different experiments

conducted in this chapter and previous one. The experiments performed in Chapter 4 compared the IRUGA performance with other GA-based IR models. The comparison is performed between different techniques of implementing GA operators, while the experiments done in this chapter compared the performance of IRUGA with traditional IR (TIR) models using several evaluation functions. The results obtained show a high improvement of IRUGA over TIR models in terms of the processing time. Applying the term proximity functions to TIR produces a high performance TPBTIR model which slightly outperforms IRUGA in terms of the three recall and precision measures defined earlier. On the other hand, applying the two well known evaluation functions, namely, the OKAPI-BM25 and Bayesian inference network models, to the TIR model results in an IR system that has a lower performance than IRUGA in terms of these three measures.

Hence, this chapter will be organized in the following order: Section 5.2 provides a description of the structure of the proposed term-proximity-based TIR (TPBTIR) model and its units as well as illustrating its performance in term of the precision and recall measures. Section 5.3 will examine the performance of the proposed TPEF when applied by the TPBTIR model. The next experiment in Section 5.4 will investigate the time performance of IRUGA and TPBTIR when both are using the TPEF function, while in Section 5.5, an experiment is conducted to compare the quality of the retrieved documents using the TPEF function, Okapi-BM25 function, and Bayesian inference network function when all are applied using the TIR approach. This experiment is conducted by examining the first retrieved document using each one of these evaluation functions and by comparing the precision and recall measures achieved by each model. The discussion of the results that have been achieved, indicating the strengths in IRUGA and TPBTIR that led to these impressive results is provided in section 5.6. This discussion includes the performance of the enhanced inverted index of IRUGA against other indexing models and the performance of IRUGA and TPBTIR in terms of the three measures. The last section, which is 6.6, summarizes and concludes this chapter.

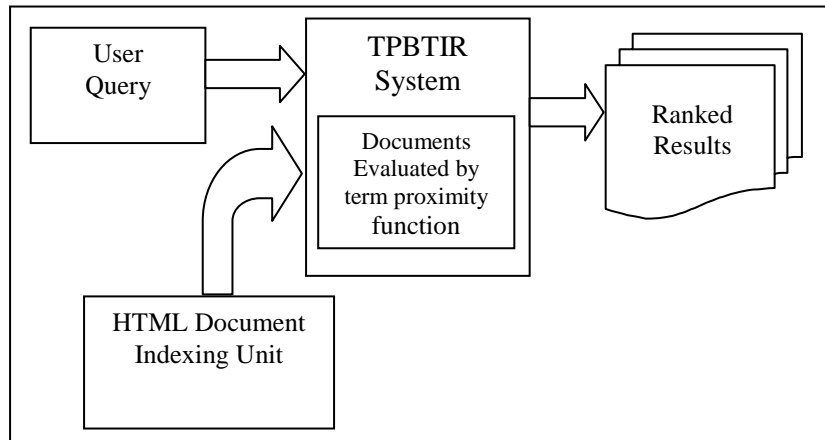
5.2 TPBTIR Structure

The Traditional IR (TIR) system typically consists of two units (Cutler, Shih, and Meng, 1999). The first unit is the *indexing tool* which extracts the useful keywords from the search

space and represents them in a way that facilitates the process of finding the relevant documents. The second unit is the *searching tool*. It compares each user query to all documents in the search space through the index database and returns a list of relevant documents. In most cases, the documents returned are ranked based on an evaluation function (Cutler, Shih, and Meng, 1999). However, the application of the evaluation functions to the set of documents returned is a crucial issue for the IR models from the computation point of view. According to (Dong et al, 2008), the extensive computation cost is a common drawback for most of the Traditional IR paradigms as it has to be applied to a vast number of documents in the data set. Nevertheless, the assumption applied by (Picarougne et al, 2002), that the user can wait a longer time (several hours) provided that he gets better results, is valid as outlined in the coming experiments as well.

5.2.1 The Design of TPBTIR

In this experiment, an example of the TIR technique is proposed and denoted as: Term-Proximity-Based TIR (TPBTIR). The indexing unit of this model is similar to that of IRUGA's. However, the mechanism of the searching unit of TPBTIR is different from IRUGA's. In IRUGA, it is the GA unit, whereas in TPBTIR, it is the proposed Term Proximity Evaluation Function (TPEF) described in Section 3.4.4. Once the user enters the query, this function will be applied to the whole document set in the search space to evaluate the documents based on this query in order to rank the results before displaying them to the user. Obviously, the evaluation of all documents in the data set is impractical, especially in the case of the web, where the number of such documents is uncountable. To further improve this, an additional criterion is applied to limit the number of evaluated documents. This criterion is to set a threshold for the number of referenced keywords within the document. If the document has a number of referenced keywords greater than this threshold then it will be evaluated, otherwise it is discarded. Consequently, only the documents that reference at least one keyword will be evaluated (if the threshold is set to one). The TPBTIR layout is depicted in Figure 5-1,



5: The TPBTIR layout

5.2.2 The Performance of TPBTIR

In this study, TPBTIR is applied to the same data set and to the same queries described in Sections 5.2 and 5.3 respectively. The threshold of minimum number of query keywords that exists in the evaluated document in this experiment is set to one. That is, if any document has at least one of the query keywords, then it is evaluated and ranked. By applying this approach, TPBTIR was able to retrieve the best relevant documents and rank them at the top position. The documents retrieved are analyzed using the measures: precision at top N, the recall and top N and precision-recall measures.

As illustrated in Table 5-1, the precision achieved by this model reaches 0.95 at first 10 documents retrieved. This implies that the number of relevant documents that appear at the first position ranges between 9 and 10 documents. Hence, the first page of results (assuming that each page displays 10 results) returns only the relevant documents. In fact, the remaining 0.05 are due to the fact that 47% of the queries have less than 10 relevant documents. In this experiment, the relevant documents appear at the top, while the irrelevant documents appear at the bottom of the resulting page. That's why the precision is very low at the bottom rows of this table. When applying the *average 11-point* precision measure, it is shown that the achieved score for this technique is 0.57. This percentage is very high when compared with the existing techniques such as (Cutler et al, 1999) and (Kim and Zhang, 2000) which achieved 0.255 and 0.2412 respectively.

Table 5-1: The performance of TPBTIR

Measure	Precision	Recall	Precision-recall
0	1.00	0.01	1.00
10	0.95	0.71	0.97
20	0.78	0.85	0.97
30	0.67	0.92	0.96
40	0.57	0.95	0.96
50	0.50	0.96	0.95
60	0.44	0.97	0.95
70	0.40	0.98	0.94
80	0.36	0.98	0.93
90	0.33	0.99	0.92
100	0.31	0.99	0.91
Average	0.57	0.85	0.95

The second measure to be considered here is the *recall at top N* retrieved documents. The importance of this measure is to show the improvement of retrieving relevant documents and to demonstrate the ability of TPBTIR to retrieve all of the relevant documents. As shown in Table 5-1, this approach is able to retrieve most of the relevant documents since this measure scores 0.99 at *recall at 100*. Knowing that 80% of the queries have less than 30 relevant documents, it is shown that the recall is more than 0.9 after the point *recall at 30*. Using the *average 11-point* recall measure, TPBTIR achieves 0.85. Once again, this value is much better than that of the existing techniques (Cho and Richards, 2004) which achieved 0.319.

The last measure to be investigated here is the *precision-recall* measure. It is one of the most common measures used to evaluate the information retrieval systems as explained in Section 5.4. This measure examines the purity of the results retrieved, since it reflects the percentage of relevant documents at multiples of 10% of the relevant documents to the total retrieved documents. The ideal case is achieved when these numbers are identical, i.e. when the number of relevant retrieved documents is equal to the number of retrieved documents or when all retrieved documents are relevant at each point in this scale.

Referring to the Table 5-1 once more, it is noted that the highest score is 0.97 at this first 10%. This means that when retrieving 10% of the relevant documents, there is a very small

fraction of irrelevant documents that are additionally retrieved. The P@R measure achieves more than 0.9 at all of the measuring points in this table, which shows that the relevant documents are always retrieved at high rank and displayed right at the top of the retrieved documents. In fact, web users are looking for such a retrieval system which saves the user from browsing many results before finding the requested piece of information. By looking at other existing techniques, it is found that (Aly, 2007) achieved precision-recall of 0.7 using the average 11-points, while (Kim and Zhang, 2000) achieved 0.7152 as an average of the first 2 points, which are precision-recall at 10 and at 20.

Comparing these results with those achieved by the existing techniques, TPBTIR proves its superiority over these models in terms of precision, recall and precision-recall measures.

5.3 Recall and Precision Performance of TPBTIR and IRUGA

The second point to be considered when comparing IRUGA with TIR is the recall and precision performance. Since TPBTIR evaluates all documents of the search space using TPEF to display the best of them in descending order to the user, TPBTIR is expected to have better performance than IRUGA for this data set using the same fitness function. The reason behind this is that IRUGA is considered as a probabilistic model; hence not all the documents in the search space are selected in the solution. This kind of selection misses some good documents, while TPBTIR examines all the documents in the search space. This allows TPBTIR to retrieve all relevant documents.

The comparison of IRUGA and TPBTIR in terms of these measures is illustrated in Table 5-2. These results are already analyzed in Section 5.6.5 for IRUGA and in the previous Section for TPBTIR, but are brought here to perform side by side comparison. In this table, the average P@N of TPBTIR is higher than that of IRUGA by 16.3%, the average R@N of TPBTIR is higher than that of IRUGA by 14.87%, and the average P@R measures are almost the same for both TPBTIR and IRUGA. These results are consonant with our expectation.

Table 5-2: Comparison between IRUGA and TPBTIR in terms of recall and precision measures

Measure	P@N		R@N		P@R	
	TPBTIR	IRUGA	TPBTIR	IRUGA	TPBTIR	IRUGA
0	1.00	1.00	0.01	0.01	1.00	1.00
10	0.95	0.86	0.71	0.63	0.97	1.00
20	0.78	0.69	0.85	0.76	0.97	0.99
30	0.67	0.57	0.92	0.81	0.96	0.99
40	0.57	0.48	0.95	0.83	0.96	0.98
50	0.50	0.41	0.96	0.84	0.95	0.98
60	0.44	0.35	0.97	0.85	0.95	0.95
70	0.40	0.30	0.98	0.85	0.94	0.93
80	0.36	0.27	0.98	0.85	0.93	0.91
90	0.33	0.25	0.99	0.85	0.92	0.89
100	0.31	0.22	0.99	0.86	0.91	0.87
Average	0.57	0.49	0.85	0.74	0.95	0.95

5.4 Time Performance of TPBTIR and IRUGA

The major advantage of IRUGA over TPBTIR is the number of documents that are evaluated upon reception of the user query before displaying the results. In TPBTIR all documents in the search space are evaluated, while in IRUGA, only the randomly selected documents are evaluated. In IRUGA, the number of evaluated documents depends on the number of chromosomes which represent the population size P and the size of the chromosome C . In a such case, this number is $P \times C \subseteq S$, where S is the search space. Adding additional criteria to the documents selected will reduce the number of these documents. The criterion applied in IRUGA is that the selected documents must have at least one keyword from the user query. Applying this criterion to IRUGA and to TPBTIR when both are using the TPEF function produces the results shown in the third column of Table 5-3, which shows slight advantage of TPBTIR over IRUGA. These figures represent the average time required by each approach using 15 queries, bearing in mind that IRUGA works in two stages. The first stage is the creation of the first generation, where the document evaluation is performed. However, most of the processing time is spent in this stage (as the few document evaluations are done during the mutation process). While the second stage, which takes a much shorter time, is the application of the GA's operators to generate the following generations and to produce the final result. Moreover, the performance of TPBTIR is much affected by the criteria of document selection. In the case

when $N \geq 0$, where N represents the number of the query keywords. In this case, TPBTIR is going to evaluate all the documents in the search space before producing the result. The second column in Table 5-3 demonstrates the performance of this scenario which shows how much faster is IRUGA than TPBTIR.

Table 5-3: Comparison between IRUGA and TPBTIR in terms of query processed time

Model	Average Time per query (sec) $N \geq 0$	Average Time per query (sec) $N \geq 1$	Average GA time (Sec) $N \geq 1$	Document Evaluation time (sec) $N \geq 1$
IRUGA	2694.2	2721.5	852	1.581
TPBTIR	5545.6	2079.6	-	1.794

5.5 The Performance of the Three Fitness Functions Using TPBTIR

This section compares the performance of the TPEF function with both OKAPI-BM25 and Bayesian inference network model functions by performing two experiments. The first experiment examines the quality or the degree of relevance of the first document retrieved by each model. The second experiment investigates the documents retrieved in terms of the recall and precision measures, which are precision at top N , recall at top N , and the precision-recall measures. The comparison between these three functions is already performed in Section 5.6.5 using IRUGA. In order to demonstrate the extreme performance of each function, the TPBTIR approach will be applied. Using this approach in this experiment rather than in IRUGA is to avoid the probabilistic behaviour of IRUGA which may miss some good documents of the search space due to the random selection of population, and to make sure that the first ranked document is the best in the search space from the evaluation function's point of view.

5.5.1 The Quality of the Documents Retrieved Using Different Evaluation Functions

In this experiment TPBTIR is tested on a set of queries listed in Appendix C. One query of this list is considered here as an example to illustrate the quality of the retrieved documents using different fitness functions, which is:

“*Mathematics experiences through image processing*”

This query exists in eight documents.

When applying TPBTIR with different evaluation functions on this query, different documents are retrieved at the first position. These documents are presented in Figures 5-2 to 5-4, where Figure 5-2 shows the first document retrieved using TPEF, Figure 5-3 shows the first document retrieved using Okapi- BM25, and Figure 5-4 shows the first document retrieved using Bayesian inference network fitness function. From these figures, one can judge that the document retrieved by the TPEF function (shown in Figure 5-2) is the most relevant document to the user query. This is because the title of this document matches the query. And the query keywords appear near the top of the document which is the title. Moreover, these terms appear in different kinds of tags, ranging from the most important HTML tag which is the *title* tag to the second most important tag which is the *header* tag ending with the least important tags which are the *bold* and *body* tag. This implies that these keywords are emphasized by the author of this document using the semantic tags (*title* and *header*) and formatting tags (*bold*).

However, the document shown in Figure 5-3, which is retrieved at the first rank by OKAPI - BM25 function, can be judged as more relevant than the one shown in Figure 5-2. This is because the percentage of occurrences of the query keywords is high compared to the size of the document. However, these keywords do not appear within other tags and are not scattered among the document. This makes the document in Figure 5-2 more relevant to this query.

The first document retrieved by the third function, which is the Bayesian inference network model, is shown in Figure 5-4. This document has many occurrences of the queried keywords. Although it references the same query terms and the same order, its content is not as relevant as the one in Figure 5-2, since the main topic of this document does not

Date: Thu, 21 Nov 1996 20:29:43 GMT Server: NCSA/1.4.2 Content-type: text/html

Mathematics Experiences Through Image Processing (METIP)

Project Director: [Steven Tanimoto](#)

[Department of Computer Science & Engineering,
University of Washington, Box 352350,
Seattle, WA 98195-2350 USA](#)

A major educational problem in United States and some other countries is that students in grades K-12 lose interest in **mathematics** and science as they progress through school. Students often complain that **mathematics** is difficult and that they don't see much use for it past simple arithmetic. In response to these concerns, the National Council of Teachers of **Mathematics** has identified a number of features that the grades 5-8 curriculum should include in order to help motivate students to stay interested:

- favor conceptual learning over rote operations;
- emphasize practical uses of **mathematics**;
- encourage discussions and group learning; and
- encourage exploratory, open-ended learning.

The goal of the METIP project is to use **digital image processing** to help meet these objectives. In particular, we have developed a series of applications designed to allow students to manipulate digitized images of their choice. These materials are intended to be used in enrichment activities rather than part of a standard classroom curriculum. Teachers can play various roles with these activities; for example, they can catalyze student learning by leading discussions of the concepts students have explored on the computer.

The METIP Project currently has a number of programs that allow students to explore **mathematics** with **image** processing:

- [The Pixel Calculator](#) (Click here to order your free copy today!)
- [The Image Wagon](#) (Click here to order your free copy today!)
- [The Transform Programmer](#) (Click here to order your free copy today!)

These applications were developed primarily for 386/486/Pentium based PC's running Microsoft Windows. One application, the Pixel Calculator, is also available for the Apple Macintosh.

Here is a list of all the [people](#) working on the METIP project.

A closely related project we are involved with is the study of [multiplayer educational activities](#). The METIP project is working to integrate the use of the WWW into its activities. Some ideas are described in [Prospects for the Direct Use of Distributed Image Databases in Educational Image Processing](#).

Currently the project is collecting the **experiences** of users with its XFORM **image** transformation software. If you have done something fun or useful with the software please let us know. We are putting the current version of its documentation online. [Here is a link to it](#). A set of slide demonstrations for XFORM has been put together by graduate students who took a [seminar during the winter of 1996](#).

Figure 5-1: The first document retrieved using the TPEF function of TPBTIR.

Date: Thu, 21 Nov 1996 22:17:29 GMT Server: NCSA/1.4.2 Content-type: text/html

CSE 590D: Special Topics

Steven Tanimoto, instructor

[CSE 590D \(Autumn 1995\): Transcript-Based Education/WWW.](#)

[CSE 590D \(Winter 1996\): **Mathematics Experiences Through Image Processing.**](#)

[CSE 590D \(Spring 1996\): **Mathematics Experiences Through Image Processing.**](#)

[CSE 590D \(Autumn 1996\): **Technology for Collaborative Learning.**](#)

Copyright Notice: The material in this course web is subject to copyright. While it may be viewed by the public, it should not be installed at any web site other than the one at the University of Washington.

This graduate seminar explores a variety of topics related to the use of computers in education. Specific topics and activities vary from quarter to quarter.
(Last Update: 30 September 1996) tanimoto@cs.washington.edu

Figure 5-2: the first document retrieved using OKAPI-BM25 function

The METIP Remote Testing Program
Project Director: Steven Tanimoto

Computer Science & Engineering Department
University of Washington, EE-35,
Seattle, WA 98195 USA

TEACH MATHEMATICS WITH IMAGE PROCESSING!

WANTED: K-12 Mathematics Teachers To Participate
in Testing Experimental Learning Materials

Digital image processing is a powerful new technology that can add an exciting dimension to mathematics classes. The project "Mathematics Experiences Through Image Processing" (or METIP) has developed experimental software and student activities with the following objectives:

- To interest students in image processing and enhance their interest in mathematics.
- To teach fundamental concepts of digital image representation.
- To make connections between mathematics and applications such as special visual effects, cryptography, image enhancement, computer vision, photography, multi-media presentation, and art.

We are looking for teachers who are willing to use these materials in their classes and help us evaluate the materials. Teachers who are enrolled in the program will receive the following benefits:

- Free copies of the software and materials. These include image processing programs and student worksheets.
- Assistance via electronic mail.

In exchange we request that enrolled teachers help us in the following ways:

- Administer the materials in a systematic manner.
- Report data and observations to us through electronic mail, or through the post ("snail mail").

Here is a description of the "course" that these materials support:

- A 5-hour course on basic digital image manipulation targeted at grades 7 and 8, but which may work well in grades 5, 6, 9, 10, 11, or 12, as well.
- A 10-hour course on basic and more advanced digital image manipulation. This is targeted at grades 8 and 9, but may be suitable for other grades in the range 7-12.

These courses require that each student have the use of a PC. The PC must be a 386, 486, or Pentium, with SVGA graphics, at least 4 MB RAM, running Windows 3.1.

Teachers will be asked to give students a questionnaire at the beginning and end of the course, and to report the following data by mail, or email: test responses, "log files" created by the software, and anecdotal information about the students' and teacher's experiences.

Timing:

The first testing phase is to take place between February 1, 1995 and April 30, 1995. Participating teachers should schedule the course to begin after February 1 and end before April 30. In a 5-hour course, students may participate one hour a day for five days (all in one week) or in any other distribution of the hours. (We are interested in finding out which arrangements work best.)

We would like to have all requests to participate by January 15. In the event that we receive more applications than we can support we may have to limit the level of support we provide to some. However, we expect to be able to handle most participants who apply.

To apply:

Please answer the following questions and press the SUBMIT button, or send the following information via email or snail-mail to the following address:

Prof. Steven Tanimoto, Project Director
 Mathematics Experiences Through Image Processing
 Dept of Computer Science and Engineering, EE-35
 University of Washington
 Seattle, WA 98195
 206-543-4848

Figure 5-3: The first document retrieved using Bayesian Inference Network function.

match the query exactly. Although all these three documents are considered as relevant to the query under consideration, it is obvious that the one retrieved by TPEF is the most relevant and reflects the accuracy of this function.

Another point worthy of attention is the evaluation value of each function. As stated in Chapter 3, for TPEF function, the evaluation score of each document has an upper limit and it is dependent on the document itself. Hence, its maximum value is one. As shown in Table 5-4, the evaluation value of the first document retrieved by TPEF is 0.7739. This gives a clue as to how relevant the document is to the query. On the other hand, the evaluation value obtained by the Okapi-BM25 and Bayesian models (41.97 and 0.826 respectively) are not bounded. Consequently, one cannot predict the degree of relevance based on these values alone unless these values are compared with the scores of the other documents in the space.

This is because they are dependent on some global factors that are obtained from all the search space. Therefore, the degree of relevance of a document is “comparative” in the sense that it is more relevant to the user query than other documents in the space.

Table 5-4 lists the statistics of the factors used by each one of these three fitness functions in addition to the rank and evaluation value of these documents according to each fitness function.

Table 5-4: Comparison of the ranks and factors of the three fitness functions for the first document retrieved by each fitness function.

Factors and Ranks	TPEF	Okapi-BM25	Bayesian
Document reference number	6371	5685	6376
Unique Terms within document	303	73	277
Document Size	624	133	579
Fitness value	0.7739	41.97	0.826
Keyword Frequency	27	10	29
No. of unique referenced keywords	5	5	5
Total Weight of all words	1664	829	1121
Rank in IRUGA	1	2	8
Rank in Okapi	3	1	2
Rank in Bayesian	2	3	1

5.5.2 Recall and precision of the three fitness functions

To overcome the randomness of selecting the documents in the initial generation creation of the GA unit of IRUGA, this experiment is performed by using TPBTIR. The advantage of TPBTIR over IRUGA is that the former evaluates all the documents in the search space and retrieves the best among the whole set, while the second approach selects a set of documents randomly and applies the operators of the GA unit to produce the best among them.

The first measure to be considered is the *precision at top N*. The results illustrated in Figure 5-5 show the advantage of TPEF over other functions. It starts by a precision of 0.95 at the top 10 retrieved documents. This means that the number of relevant documents within the first 10 retrieved documents ranges between 9 and 10. A few of the queries have the number of relevant documents less than 10, causing this percentage to be below one. This score decreases until it reaches 0.31 when retrieving 100 documents. That means, within the top

100 retrieved documents, only 31 of them are relevant. The reason behind this low percentage at P@100 is either that the number of relevant documents is small, so the relevant documents are displayed at high rank, or this function is not able to rank the relative documents at high position as it assigns them a low evaluation value, thus causing some relevant documents to be retrieved at very low rank. Knowing that the average number of relevant documents per query is 19.24, the first option is more likely to be the actual reason for this percentage. In general, the average improvement achieved by TPEF is 37.21% over OKAPI-BM25 and 84.81 % over Bayesian as shown in Table 5-5.

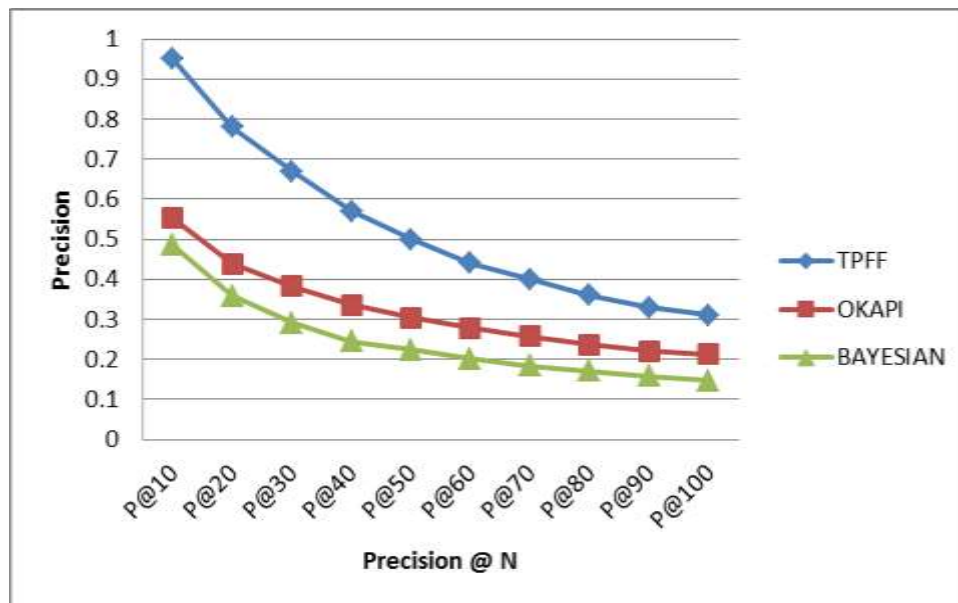


Figure 5-4: The precision improvement of TPEF over other functions

The second measure to be analyzed is the recall at top N. This measure shows how many relevant documents are retrieved at the top position of the displayed results. The aim is to retrieve all the relevant documents and to display them at the top. TPEF managed to achieve 99% of this aim by retrieving 99% of the relevant documents as depicted in Figure 5-6 and numerically represented in Table 5-6. Knowing that 2% of the queries have more than 100 relevant documents justifies the reason why not 100% is achieved here. However, the improvement of TPEF is noticeable where it ranges between 16.44% when compared with Bayesian inference network, and 18.87 % when compared with OKAPI-BM25.

Table 5-5: Percentage of precision improvement of TPEF over other functions

Measure	TPEF	OKAPI	% of Improvement	Bayesian	% of Improvment.
P@0	1.00	1.00	0	1.00	0
P@10	0.95	0.64	49.46	0.54	74.63
P@20	0.78	0.50	54.79	0.40	94.22
P@30	0.67	0.44	52.11	0.33	105.92
P@40	0.57	0.39	47.76	0.27	108.33
P@50	0.05	0.35	43.02	0.25	99.81
P@60	0.44	0.32	37.25	0.23	95.53
P@70	0.40	0.30	35.41	0.21	93.53
P@80	0.36	0.27	32.59	0.19	87.16
P@90	0.33	0.25	29.74	0.18	86.38
P@100	0.31	0.24	27.15	0.17	87.38
Average	0.57	0.43	37.21	0.34	84.81

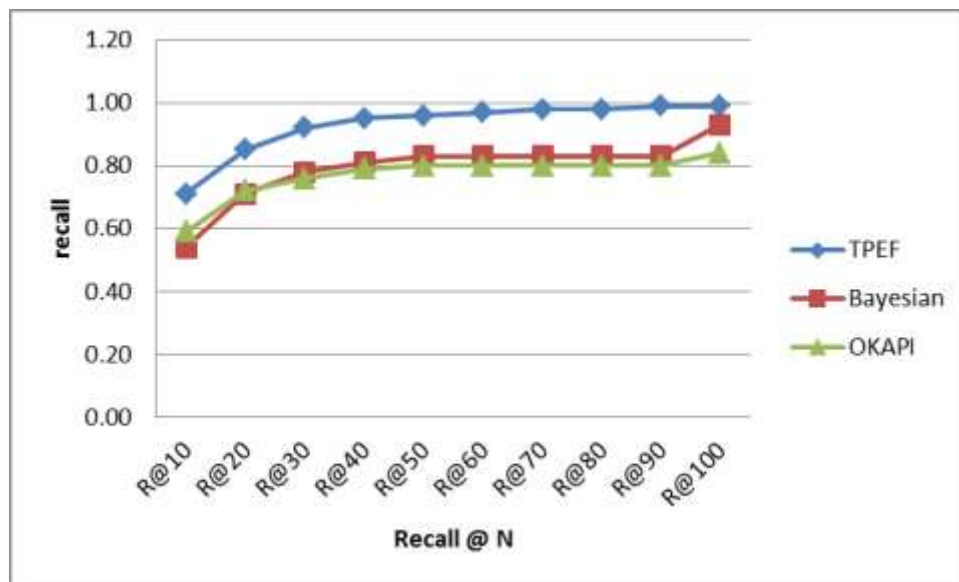


Figure 5-5: The recall improvement of TPEF over other functions

The last measure to be investigated is the precision-recall measure. It computes the total number of documents retrieved when retrieving multiples of 10% of the total relevant documents. That is, in order to retrieve the first 10% of the relevant documents, how many additional irrelevant documents are retrieved? Figure 5-7 shows that all three functions start at good points that are between 0.96 and 0.97 for P@R10. But when retrieving more

Table 5-6: Percentage of recall improvement of TPEF over other functions

Measure	TPEF	Bayesian	% of Improvement.	OKAPI	% of Improvement.
R@0	0.01	0.01	0	0.01	0
R@10	0.71	0.54	31.48	0.59	20.34
R@20	0.85	0.71	19.72	0.72	18.06
R@30	0.92	0.78	17.95	0.76	21.05
R@40	0.95	0.81	17.28	0.79	20.25
R@50	0.96	0.83	15.66	0.80	20.00
R@60	0.97	0.83	16.87	0.80	21.25
R@70	0.98	0.83	18.07	0.80	22.50
R@80	0.98	0.83	18.07	0.80	22.50
R@90	0.99	0.83	19.28	0.80	23.75
R@100	0.99	0.93	6.45	0.84	17.86
Average	0.85	0.72	16.44	0.70	18.87

relevant documents the performance drops gradually for both Bayesian and OKAPI functions, while the performance of TPEF drops slightly until it reaches 0.91 when retrieving all the related documents as illustrated in Table 5-7. That means, when retrieving all the related documents only 9% of the retrieved documents are irrelevant, and these irrelevant documents appear at the bottom of the displayed results. The results achieved are very near to the user's expectation since he or she is expecting all top ranked documents to be relevant, and most of the relevant documents appear in top position.

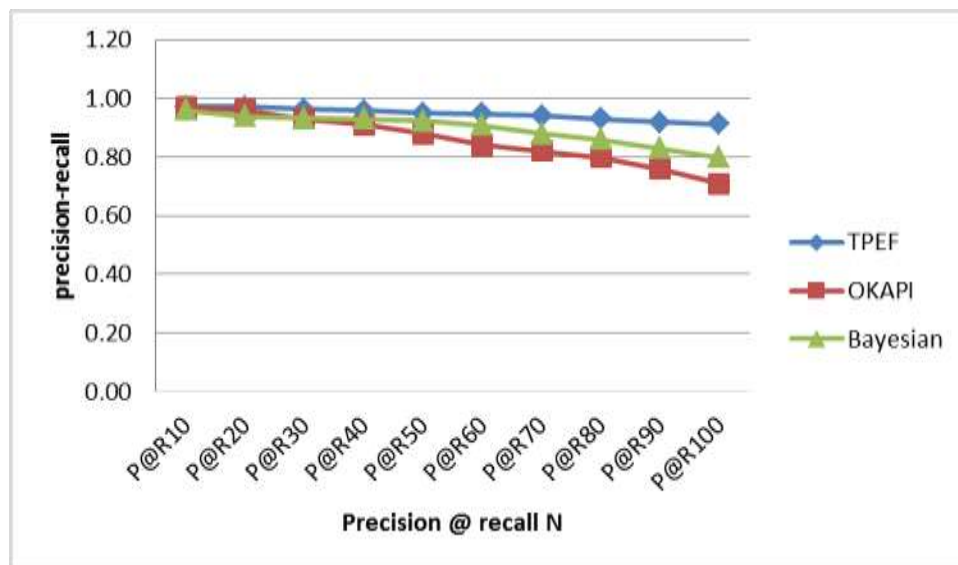


Figure 5-6: The precision- recall improvement of TPEF over other functions

Table 5-7: Percentage of precision- recall improvement of TPEF over other functions

Measure	TPEF	OKAPI	% of Improvement.	Bayesian	% of Improvement.
P@R0	1.00	1.00	0	1.00	0
P@R10	0.97	0.97	0.27	0.96	1.31
P@R20	0.97	0.96	1.25	0.94	3.40
P@R30	0.96	0.93	3.51	0.93	3.23
P@R40	0.96	0.91	5.36	0.93	3.09
P@R50	0.95	0.88	7.98	0.92	2.93
P@R60	0.95	0.84	12.72	0.91	4.11
P@R70	0.94	0.82	14.80	0.88	6.98
P@R80	0.93	0.80	16.40	0.86	8.28
P@R90	0.92	0.76	20.91	0.83	10.71
P@R100	0.91	0.71	28.61	0.80	14.14
Average	0.95	0.87	10.16	0.91	5.29

5.5.3 IRUGA Vs TIR Using the Three Evaluation Functions

To summarize the experiments performed so far in this chapter, the results are presented together in Table 5-8. This table shows that the best performance achieved is when using the TPBTIR model. This model achieves a best precision of 0.57, a highest recall of 0.85, and a highest precision-recall of 0.95. This approach achieves the best performance due to the combination of the high performance evaluation function and the natural behaviour of the TIR which parses the whole document set in order to obtain the best document among the data set. However, this high achievement is accomplished at the expense of the time spent by the user waiting for the system to evaluate every single document to get such results.

Table 5-8: Summary of TPDTIR and IRUGA performance

Model	Evaluation Function	P@N	R@N	P@R
TPBTIR	TPEF	0.57	0.85	0.95
TIR	Bayesian	0.34	0.72	0.9
TIR	OKAPI-BM25	0.43	0.70	0.87
IRUGA	TPEF	0.49	0.80	0.95
IRUGA	Bayesian	0.25	0.71	0.91
IRUGA	OKAPI-BM25	0.32	0.68	0.85

IRUGA is a good alternative to TPBTIR when the processing time is concerned, and produces precision-recall that is very close to TPBTIR's. Although its performance is slightly lower than TPBTIR in terms of P@N and R@N measures, the results are still relatively high and much better than those of the OKAPI-BM25 and the Bayesian inference

network models.

5.6 Discussion

This section discusses the results obtained by different experiments conducted in the previous chapter and this chapter. The experiments performed in Chapter 4 compare IRUGA performance with other GA-based IR models. The comparison is performed between different techniques of implementing GA operators, while the experiments done in this chapter compared the performance of IRUGA with TIR models using several evaluation functions. The results obtained show a high improvement of IRUGA over TIR models in terms of the processing time. Applying the term proximity functions to TIR produces a high performance TPBTIR model which slightly outperforms IRUGA in terms of the three recall and precision measures defined earlier. On the other hand, applying the two well known evaluation functions, namely, the OKAPI-BM25 and Bayesian inference network models, to the TIR model results in an IR system that has a lower performance than IRUGA in terms of these three measures.

This section will discuss the results that have been achieved, indicating the strengths in IRUGA and TPBTIR that led to these impressive results, the performance of the enhanced inverted index of IRUGA against other indexing models, then finally discusses the performance of IRUGA and TPBTIR in terms of the three measures.

5.6.1 The Performance of the Enhanced Inverted Index

Both IRUGA and TPBTIR are built on top of the enhanced inverted index. For each indexed term, this index stores a list of documents referencing it in addition to its position within the document and within the sentence. Moreover it stores the HTML tag weight for these terms. This indexing model gives the flexibility to associate as much data as possible to the term without much affecting the storage space or retrieval speed. In contrast, vector space requires a huge amount of space (Snasel, Moravec and Pokorny, 2005), expressed as $n \times m$, where n and m are total number of terms and total number of documents in the search space respectively. In Section 5.5 it is shown that this indexing technique saves 99.5% of the space that would be used by the vector space. This finding supports the main drawback of the vector space model mentioned in (Snasel, Moravec and Pokorny, 2005), which states that

the document vector has a big dimension and requires huge storage volume if stored as classical vectors. Moreover, this type of indexing increases the load on the system resources by increasing the computation cost during the document evaluation process (Dong et al, 2008). Latent semantic indexing (LSI) was another option to use as an indexing model. However, the data set used in this research consists of 128213 unique words distributed over 8344 documents. Applying LSI to this data set reduces the noise and removes the redundancy from the indexed terms (Dong et al, 2008). However, constructing this index consumes a longer time since it will parse whole documents to build the initial vector space and obtain the singular value decomposition before constructing another compressed semantic version of the vector space matrix that includes only the important terms and removes the noise and redundant terms. In addition, this model has the same drawback as the vector space, that is the limitation of storing additional data associated with each term.

5.6.2 Performance of IRUGA and TPBTIR

Several experiments are applied to IRUGA and TPBTIR to compare their performance with the existing techniques. In this section, IRUGA and TPBTIR will be compared with GA-based IR approaches as well as other TIR approaches. Despite the huge amount of research that is investigating IR performance, it is found that only a few of them mention explicitly the numerical results of precision, recall and precision-recall achieved by their techniques. Hence, this chapter will focus on these approaches and compare them with IRUGA and TPBTIR. Moreover, from the GA point of view, since the research in literature on GA approaches focuses on one or two operators only per study, it is difficult to compare the performance of IRUGA with each existing technique. Hence, the comparison will be based on the evaluation measures mentioned in Section 4.4 and the techniques applied by these studies are highlighted.

5.6.3 Convergence speed

This measure is applicable for GA-based approaches; hence, TPBTIR will be excluded from this comparison. One of the objectives of IRUGA is to enhance the speed of retrieving the results. This speed can be controlled in two ways. The first one is to speed up the process of creating each generation. This is mainly affected by the complexity of the fitness function

and the crossover operator in addition to the processor speed. Since this measure is not mentioned by other research work it will not be considered in this study.

The second method of measuring the speed is to minimize the number of generations, or in other words: to tune the GA operators in order to speed up the convergence. Although this measure is not tackled explicitly by researchers, it is pointed out as an advantage of a specific technique that allows fast convergence (Yang, Korfhage and Rasmussen, 1992; Pathak, Gordon and Fan, 2000). In Section 4.7.1, a comparison is done between different techniques of GA operators and is summarized in Table 4-29. These figures are obtained from the experiments applied to IRUGA when compared with other techniques. Since such results are not mentioned explicitly by other researchers, it is going to be induced from their work.

In literature, the number of generations is mentioned as a parameter of the GA model where it ranges between 20 (Kim and Zhang, 2000) and 500 (Vrajitoru, 2007). However, few of them include the number of generations in the results graphically (Losee, 1996) or in tabular form (Vrajitoru, 1997). (Losee, 1996) represented the results graphically where the maximum number of generations was 90 for the first experiment and 60 for the second one, whereas the number of generations in (Kim and Zhang, 2003) is 30 when examining the HTML tag weight using GA. These studies show that IRUGA converges faster. On the other hand, (Kim and Zhang, 2000) plotted the results of average fitness per generation and show that the maximum number of generations is 20 which somehow indicates the speed of convergence for this model. This is slightly better than IRUGA's, but its quality of the results in terms of precision and recall are lower. Nevertheless, there is no special importance given to this factor. Therefore these figures will not reflect the actual speed of convergence but can provide an idea about where IRUGA fits for this measure. (Húsek et al 2005) produces recall of 1 when the number of generation is very huge where it is 1200. While (Radwan et al, 2006; Aly, 2007) converges within 100 generations. Much apart from these is the number of generations required for (Marghny and Ali, 2005) approach to divert which is 12000 generations. Of course, this gives an idea about the slow performance of such approach which aims to produce high average mean quality of the retrieved documents. And the achieved quality is 25%.

The average number of generations formed by IRUGA was 22.26. The main reason of the combination of the low number of generation and high precision and recall is the way the IRUGA's hybrid crossover is implemented. As illustrated in Table 4-5, this crossover technique is faster than non-ordered crossover by 20.5% and faster than one-point crossover which produces two offspring by 48.71%. However, it was slower than 2-point crossover by 63%. However, the later performance was very poor in terms of recall and precision as shown in Section 4.7.4.1. The reason is that this technique pushes the high quality genes towards the left of the chromosome by ordering the genes and by combining the best genes of both parents into one offspring.

5.6.4 Precision at top N (P@N) Measure

The average precision for the 11-point measure achieved by TPBTIR and IRUGA are 0.57 and 0.47 respectively when considering all queries. However, this measure is heavily affected by the number of relevant documents per query, and here is the explanation: P@0 means the precision at first retrieved document and IRUGA always retrieves a relevant document at the first position, so IRUGA doesn't have a problem at this point. The issue is in the following points. P@10 is considered as an example. This measure represents the percentage of the relevant documents within the first 10 retrieved documents. If the total number of relevant documents is less than 10 then this value will always be less than 1. But if the total number of relevant documents is greater than 10, then this value can reach 1 if all top 10 retrieved documents are relevant. Since 53% of the queries used by IRUGA have a relevant number of documents of less than 10, then there is a probability of 53% for P@10 not to reach 100%. Thus, this measure can be evaluated again for the documents having $N > 10$. In this case, P@10 jumps from 0.85 to 1 and the average precision jumps from 0.49 to 0.62. Similarly, this measure jumps from 0.95 for TPBTIR to 1 at P@10 and the average precision jumps from 0.57 to 0.71.

From the literature, it is shown that many researches apply the common fitness functions which is TF-IDF. In fact, this function is based on statistical factors only, namely, term frequency, number of documents referencing this term and total number of documents in the search space. Moreover, this function favours the documents with high term frequency regardless of the document size since the second factor (IDF) is constant for all documents

under consideration. To overcome this drawback, Vrajitoru used the normalized form of TF-IDF by dividing each component by the document size. So the document is now evaluated based on the percentage of the term frequency within the document. In general, still this formula not enough to completely evaluate the document. When applying this formula to HTML documents, (Cutler et al, 1999) introduced an additional vector called: Class Importance Vector *CIV* concept. The values within this vector represent a weight assigned to the HTML tags under consideration. This vector is multiplied by the term frequency vector *TFV* in order to measure the relevance of a document.

(Kim and Zhang, 2003) has adopted GA to evaluate the weight to be assigned for HTML tags and applied these weights to three know evaluation functions, namely, TF-TDF, 2-poisson model, and Bayesian inference network model, (Refer to table 7-1 for listing of these formula). Among these three formulas, the best result is achieved by the 2-poisson model, and it was 0.63 for P@10 and 0.46 for P@20. In spite of that, these results are still far from the results achieved by IRUGA, namely, 0.86 and 0.69 for the same measures.

An example is presented in this context to compare the quality of documents retrieved by TPDF with that retrieved by TF-IDF. This example is applied on query number 7 in Appendix C. This query is: “*caltech computer science department*”. The first document retrieved by both IRUGA and TF-IDF model are depicted in Figure 5-8 and Figure 5-9. (For simplicity they are referred as D1 and D2 respectively). D1 has the 4 keywords of the query, while D2 has only 3. D2 is retrieved in position 564 in IRUGA model since it has very low relativity (score of 0.3284 using TPDF), while document1 come in position 14 using TF-IDF model. Details of calculating TF-IDF for D2 is presented in Table 5-9, where N is the total number of document in the space which is 8349. This example proves one of the major drawbacks of TF-IDF and its variations since it concentrates on the popularity of the keywords among the document set as it favours terms that are less referenced and gives them higher weight. Once again this is considered as a global factor, and makes the relevance depending on the data set rather than depending on the content of the document itself.

5-9: Factors used to calculate TF-IDF for document 2

WORD	N (total document referencing the term)	log(N/df)	freq.	TF-IDF /word
CALTECH	142	1.7693461	77	136.239651
COMPUTER	4494	0.2690014	5	1.345006965
DEPARTMENT	2911	0.4575923	0	0
SCIENCE	3612	0.3638867	6	2.183320316
Document TF-IDF				139.7679783

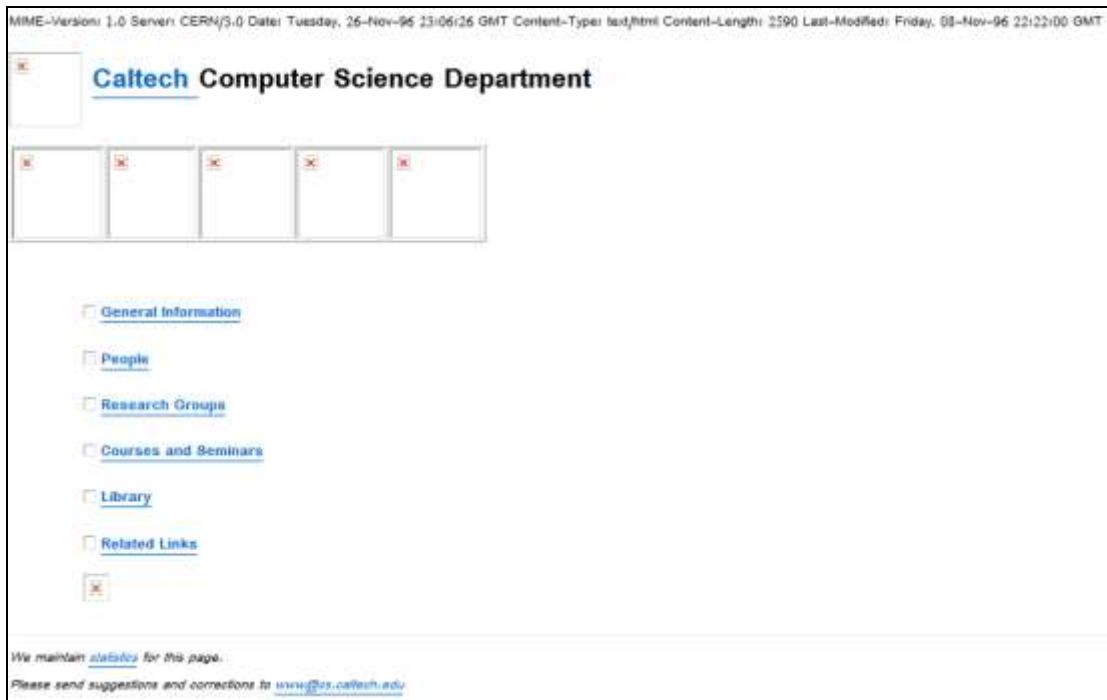


Figure 5-7: The first doc retrieved by IRUGA for query 7 of table 7-2.

From another prospective, it is shown that the documents presented using the vector space model are evaluated using TF-IDF, or other well known similarity measure which are compatible with the vector representation such as cosine measure (Aly, 2007), Jaccard coefficient and Dice coefficient (Klabbankoh and Pinngern, 2008; Saini, Sharma, and Gupta, 2011) (refer to Table 7-1 in appendix A for the formulas of these models). This representation may produce reasonable results according to the mathematical evaluations where (Klabbankoh and Pinngern, 2008) achieved maximum precision of 0.746 and (Saini, Sharma, and Gupta, 2011) achieved precision of 0.705. However, these formulas are restricted to statistical factors which depend on the density of the terms within the document itself or within the collection. Similarly is the probabilistic model in which the probabilistic

theories are applied to evaluate the document based on the entered keywords. The approach proposed by (Zhang, Wei and Meng, 2012) applies the probabilistic information retrieval model to capture the correlations between the unspecified and specified values of leaf nodes as well as the user preferences based on the XML data and query history, where leaf nodes corresponds to product details. This model is applied on used car dataset consisting of 100000 car parts and achieved average precision of 0.79. However, this approach is more suitable for question-answering systems than general document retrieval system.

Moving further toward other types of factors used to evaluate the documents it found that (Cummins and O’Riordan, 2006) has categorized the factors into local factors and global factors, and proposed an evaluation function (formula 6 in table 7-2) that is composed of all of these factors, then compared it with both TF-IDF and BM25 formulas. This evaluation function performs better than them but with maximum average precision of 0.588 on a collection of 1033 documents. However, this document set is not big enough to prove efficiency.

Another factor to be considered in evaluating the document is the term proximity. (Tian et al, 2006) applied several term proximity concepts in evaluating the document. These concepts include Mimum Term Distance (MTD), First Appearance of MTD (FAM) and Local Appearance Density LAD. This technique is applied on 20 documents and found that the precision ranges beteen 0.85 for P@1 and 0.4 for P@20. This technique shows enhancement in the precision compared with above techniques. The best improvment is achieved by MTD concept. Hence it is adopted in this work as part of the proposed fitness function.

Applying the HTML tags, it is found that (Cutler et al, 1999) and (Kim and Zhang, 2000) have achieved precision of 0.254 and 0.2412 respectively. These two approaches are applied to HTML documents using the vector space model for document representation. GA was applied in these systems to find the best HTML tag weight. In (Cutler et al, 1999), the results are mainly controlled by the crossover technique in addition to the method of evaluating individuals. The crossover is done for each consecutive pair of the parent population with a probability of 0.75 and it is performed using a binary mask to exchange the 1’s bits of parent chromosomes. However, using binary masks for crossover causes the

Date: Thursday, 21-Nov-96 22:09:22 GMT Server: FQDN/1.3 MIMM -version: 1.0 Content-type: text/html

Bryan Chow's Home Page

Welcome to my World Wide Web Home Page! I am a graduate student at the [California Institute of Technology](#). I am working towards my PhD in [Computer Science](#).

Quick Go To: [Cars and Driving](#) [Roxy Music](#)

I work at the [Stanford Concurrent Programming Laboratory \(SCP\)](#), under Professor Steve Taylor.

[X](#) [X](#)

Getting in contact with me:

Email:

- Main account: bryw@psc.sulstatk.edu or bryan@cs.sulstatk.edu
- Stanford account: bryw@cs.stanford.edu
- MIT Artificial Intelligence Lab account: bchwo@ai.mit.edu

Phone:

- (818) 207-2803 Work
- (818) 795-1819 Home
- (818) 792-4157 Fax

Snailmail:

- Oncampus:
Bryan Chow
Computer Science 338-XD
Caltech
Pasadena, CA 91125
- Residence:
Bryan Chow
1127 Del Mar Blvd, #211
Pasadena, CA 91106

We have a few video cameras in our lab (on our SGI machines), and [Mits](#) has set them to take pictures of our lab every five minutes. Here are the most recent ones:

[X](#) [X](#) [X](#)

Education

- [California Institute of Technology](#): Candidate for PhD in Computer Science.
- [Stanford University](#): Graduated in June, 1994. Masters in Computer Science.
- [University of California at Berkeley](#): Graduated in December, 1992. Bachelor of Science in Electrical Engineering and Computer Science.

Work Experience

Between January and September 1993 I worked as a software engineer for a cool company called [Networks](#) which produces the [OS/2](#) operating system. Among the products are the [Gosoft StarWriter](#) and the [Gosoft/Tandy Source PDA](#). I also wrote [Blackjack](#) and [Poker](#) for the [Quick Stuffz Game Pack](#).

Project

I am currently working on a project with the [MIT Concurrent VLSI Architecture](#) group. Specifically, I am working on Multi-cluster compilation on the [M-Machine](#).

My Other Pages

These are some other pages that I maintain.

- [Cars and Driving](#) - My auto page
- [Madness in My Soul](#) - The Roxy Music and Bryan Ferry Homepage
- [Photo Album](#) - Pictures of my family
- [Fav Music](#) - Some of my favorite albums

Miscellaneous Links

- [The Complete Works of William Shakespeare](#)
- [eMail](#) - The Information Superhighway Mail!
- [Gilbert Architects](#)
- [Fun and Games at Caltech](#)
- [Pet Shop Boys](#)
- [Hates MIT](#) - My high school in Singapore has its own homepage!
- [Yahoo](#) - A Guide to WWW

World Wide Web Search Box

Find all matches containing all the following keywords(separated by space):

This is the 6952nd access of this page since June 23rd, 1995.

Bryan Chow: bryw@psc.sulstatk.edu

[California Institute of Technology, Pasadena, CA 91125](#)

5-8: The first doc retrieved by TF-IDF formula for query 7 of table 7-2

destruction of the building blocks as explained in Chapter 2. The results of this approach show the advantage of using the HTML tags over the plain text in evaluating the document using this measure. Since chromosomes in IRUGA are implemented using the ordered technique, where the genes are order from highest to lowest performance, the binary mask crossover is not suitable for it, because the offspring will be constructed by selecting random genes from all location from the parent, knowing that the weak genes are concentrated on the right side of the chromosome, then there is probability of 50% that some of these will be inherited to the offspring which will reduce its performance.

The advantages of TPFf over aforementioned formulas can be summarized in the type of factors considered in evaluating the document and their formalization. One of the factors that feature TPFf from all of the above formulas is the percentage of query keywords that exist within the retrieved document. This feature is given high weight among other factors. Its effect is clearly notified by the above example. Another feature of TPFf is the ability to determine the degree of relevance of the document to the query independently from other documents in the space. Moreover, IRUGA considers the term distance and assigns a higher score for documents having shortest distance between query keywords in addition to considering the position within the HTML tags and the location within the document.

However, IRUGA applied the hybrid crossover. This technique generates one offspring per crossover operation which minimizes the chance of delaying the convergence since the best genes are grouped together using the one-point crossover and ordered within the chromosome. In addition, this technique avoids falling into local optima.

In fact, the performance of IRUGA is not tested against multiple data sets, but as the evaluation of document is done independently of other documents in the set, then it is expected to have consistent performance regardless of the data set type as long as the document type is HTML documents.

5.6.5 Recall at top N (R@N) Measure

Few authors used this measure to evaluate their GA approach. However, it is introduced by Cho and Richards (2004). Their technique applied Formal Concept Analysis to Web documents which are related to a specific domain. In this technique, the system builds a

concept map tree based on the user queries. Upon the user query, it checks the Concept Tree Map (CTM) against the query. Based on the query, the system provides the user with information. If the user decides that the result is unacceptable, the system then proposes a list of concepts related to the user's query. If the list still does not include acceptable results, the user adds a new concept. The high performance of this system is achieved when it is applied to a narrow domain. One of the measures used to evaluate this approach is R@N. It achieved a maximum average of 0.319. However, this score is much less than that of IRUGA and TPBTIR which achieved average R@N of 0.74 and 0.85 respectively. Although (Klabbankoh and Pinngern, 2008) achieved high recall of 0.976, it uses very simple implementation technique, where the data set is set of 5 documents, each is composed of small sentence (less than 10 words). It applied vector space model and binary representation to compare between Dice coefficient, Cosine coefficient and Jaccard coefficient. Hence then technique is not robust enough to be compared with IRUGA. (Alzahrani and Salim, 2009) achieves recall of 0.7 when using the fuzzy IR rather than the GA approach. It is applied on a set of 500 Arabic documents using 15 queries. This approach indexed the terms within the document by building unique term pairs. For each pair of terms, a term-to-term correlation factor that defines the extent of relevance between these two pairs is computed. Then the documents are evaluated by measuring the degree of similarity between the document under consideration and the query document. Although this model is tested on HTML document set, but the author treat the HTML as stop words. In addition, this approach constructs a term-to-term correlation matrix thesaurus which includes all unique pairs of terms extracted from the document set and the document query set. As mentioned by the author that one disadvantage of this technique is that it consumes long time and large space to implement. Moreover, still its performance is behind IRUGA's by 5.7%. As a conclusion, IRUGA achieved minimum enhancement of recall by 5.7%.

5.6.6 Precision at Recall (P@R) Measure

Many studies investigated the precision at recall measure for their GA approaches. The maximum value is always achieved at 10%. In addition, some studies (Vrajitoru, 1998; Cutler et al, 1999; Kim and Zhang, 2000; Horng and Yeh, 2000; Aly, 2007) used the 11-point measure for evaluating the results obtained while others represent the results at fewer

points such as 10% of P@R (Kim and Zhang, 2000). Therefore, IRUGA will be compared with the existing techniques based on the evaluation measure used for that technique.

Just to remind the reader that both IRUGA and TPBTIR achieved a score of 1 at 10% of P@R and 0.95 on average on the 11 points of the P@R measure.

Vrajitoru (1998) built a GA approach using the vector space model and evaluates the documents using the normalized version of classical TFIDF formula: $ndf \bullet nidf$. This approach is characterized by a new technique for crossover called dissociated crossover. In this technique, two offspring are generated differently from the same two parents using 2-point crossover, but the genes between the two crosspoints are treated differently. The first offspring is generated by the simple 2-point crossover, while in the second offspring, the genes are replaced by 0's. However, replacing these locations by zeros reduces the performance of the chromosomes. The effect of such a technique is examined against the precision of two data sets: CACM and CISI. This technique was evaluated using the non-interpolated average precision-recall of 0.429 for CACM, and 0.2182 for CISI data set. This is lower than that of IRUGA's by 96.6%, which implies that dissociated crossover technique is not suitable for all data sets as stated by the author, and it produces high diversity of performance depending on the data set.

Another technique to be discussed is the one developed by Kim and Zhang (2000). This technique managed to achieve 0.7152 at 10% of P@R. This implies that there are many impurities in the top ranked retrieved documents. In fact this score means that to retrieve 10% of the relevant documents there are around 30% within this 10% that are not relevant. Having this percentage appearing at the top ranked retrieved documents reflects the weakness of such an approach. Consequently, the 11-point measure for P@R of this approach is relatively as low as 0.286.

Close to the results of this study is the approach developed by (Aly, 2007). This approach uses the vector space model in representing the documents and applies the cosine similarity function as a fitness function. The crossover technique applied is the simple one-point crossover, where the genes are exchanged between the parents after the crosspoint to produce the offspring. This approach achieved 0.297 for the 11-points P@R which is close to the results achieved by the above Kim and Zhang (2000) approach. These results reflect

the low performance of these techniques.

Horng and Yeh (2000) used GA as part of their IR model to tune the weights of keywords. This model uses the vector space and was applied to plain Chinese document in addition to applying relevance feedback from the user. This technique produces one offspring from the crossover operator. Two types of crossover technique are applied. The first one is the weight-selection which uses random recombination from the parent classifier, and the second one is Natural crossover where the offspring Q^z is generated from parents Q^x and Q^y such that $Q^z = (w_1^z, w_2^z, w_3^z, \dots, w_i^z)$, where $w_i^z = w_i^x + (w_i^z - w_i^y)$; $1 \leq i \leq n$ and w_i is the weight of the keyword k_i . This technique produces offspring which have similar or better performance than the best parent. Several experiments are conducted in this study, but the best results achieved an average of 0.7003 using the P@R measure. The IRUGA's performance is better than this technique by 65.6%.

Another approach that applies this measure is developed by Desjardins, Godin, and Proulx (2005). They represent the documents by "concepts". These concepts are defined by the sets of correlated terms rather than by raw terms. In their approach, they used GA to discover the best sets of co-occurrent terms, and then investigated the results when applied to IR. This study didn't mention the average P@R explicitly; rather, it only presented the results of a precision-recall curve through a diagram. From this diagram, it is noted that the maximum P@R achieved is 0.225 compared to maximum of 1 for IRUGA which means that the enhancement of this measure achieved by IRUGA is 344%

Considering 9-points from 0.1 to 0.9, (Radwan et al 2006) applied their GA technique to 3 data sets and get relatively high results compared with others. They tested their technique on CISI, CACM and NPL data sets which range in size between 1460, 3204 and 11429 documents. The results obtained are 0.437, 0.334, and 0.401 respectively. However, this approach used the plain text type of documents which are indexed by the vector space model. The terms are weighted using the Salton and Buckley formula (formula number 10 in table 9-1). The evaluation of documents is done using the fitness function which evaluates the difference between term weights of a given chromosome and the query vector. This approach differs from IRUGA by the document type and document representation.

Hence, this approach follows the algebraic category based on the TIR classification explained in Chapter 2. The IRUGA's performance is better than this technique by 117.4%.

One of the approaches that use precision-recall measure combines GA with simulated algorithm based on the vector space model (Xu, Deli and Yu, 2009). The maximum achieved score for precision-recall measure is 0.75 for P-R@10 and drops to 0.6 for P-R@100. This approach is implemented using VSM and classical TF-IDF evaluation function to cluster web documents in order to improving web mining. This approach faces the shortage of vector space and TF-IDF explained earlier. And its performance is lower than IRUGA by 33%.

Last approach to be discussed here is the one developed by (Saini, Sharma, and Gupta, 2011). This approach proposes a new semantic based similarity measure in which each phrased term is or single term word is assigned a weight based on its semantic importance considering. Different similarity measure are applied such as semantic similarity measure, Jaccard and cosine to form the semantic-based-combined-similarity-measure. This approach is implemented using GA with classical operators (crossover with probability of 0.6 and mutation with probability of 0.02. and Roulette wheel method for parent selection). It is noted that the precision-recall achieved is 0.932 for P-R@10 which is very good score compared to other IR systems investigated above, but the performance drops dramatically to 0.19 for P-R@100 when retrieving all relevant documents, compared with IRUGA that achieves 0.87. Hence, IRUGA achieves an enhancement of about 69.3% over this approach.

The summary of the performance of the IR techniques discussed in terms of the three measures is presented in table 7-1, where the techniques are classified as GA based (GA) and Non-GA-based (NGA) which can be one of the categories described in Section 2.3. The first two rows show the superior results of both IRUGA and TPBTIR when both are applying the TPEF function using the 11-point measure. The performance of the three measures is enhanced by more than ranges between 5.7% and 344%. The performance of IRUGA for P@10 and P@20 (N<3) are also mentioned to be compared with the approach proposed by (Kim and Zhang, 2003), since this study represents the results at these points. In this table, it is noted that the approach developed by (Sehgal et al, 2009), has the highest P@N score (0.823). In fact this study is implemented on plain text using the VSM indexing

model and applies the feature value version of TF-IDF. Moreover, it does not mention the window size to assure its compatibility to our comparisons.

The novelty of IRUGA appears in the measure of precision-recall, where it achieves a score of 0.95 that is not achieved by any other technique as per the investigated literature. This represents the purity of top retrieved documents, where all relevant retrieved documents are appearing at the top rank.

Table 5-10: Summary of the performance of the IR techniques discussed

Model	Classification (GA/NGA)	Notes	P@N	R@N	P@R
TPBTIR	NGA	TPEF; 11-points measure	0.57	0.85	0.96
		N<3	0.865		
IRUGA	GA	TPEF; 11-points measure	0.49	0.74	0.95
		N<3	0.775	0.32	1
		N=1	0.86	0.63	1
		N=10	0.22	0.86	0.87
Vrajitoru, 1998 (ACAM data set)	GA	11-points measure			0.415
Vrajitoru, 1998 (CISI data set)	GA	11-points measure			0.218
Cutler, et al, 1999	NGA	11-points measure	0.255		
Kim and Zhang, 2000	GA	11-points measure	0.241		0.286
Vrajitoru, 2000 (CISI data set)	GA				0.383
Hornng and Yeh, 2000	GA	11-points measure			0.7
Kim and Zhang, 2003	GA	N<3	0.545		
Cho and Richards, 2004	NGA	window size not mentioned	0.685	0.371	
Desjardins, Godin, and Proulx, 2005	GA	maximum value achieved			0.225
Húsek et al 2005	GA	11-points measure	0.75	1	

Radwan et al, 2006	GA	Interpolated measure			0.437
Aly, 2007	GA	11-points measure			0.297
Yeh et al, 2007	GA	N=1	0.58		
		N=10	0.2		0.19
Klabbankoh and Pinnjern, 2008	GA	$P_{mute}=0.01$	0.746		
		$P_{mute}=0.3$		0.976	
Manning, Raghavan, Schütze, 2009	NGA	11-points measure	0.246		
Xu, Deli and Yu, 2009	GA	N=1			0.75
Yan et al, 2009	GA	N=1			0.9
		N=10			0.07
Alzahrani and Salim, 2009	NGA	11-points measure	0.705	0.7	
Sehgal et al, 2009	NGA		0.823	0.857	
Dashti and Zad, 2010	GA				
Penev and Wong, 2010	NGA	P@3	0.6		
Saini, Sharma, and Gupta, 2011	GA	N=1			0.932
		N=10			0.19

5.7 IRUGA Vs Commercial Search Engines

This section discusses the factors applied by IRUGA to retrieve the documents and compares it to that of the commercial search engines. The factors used by IRUGA to retrieve a document are explained in section 3.4.5. Actually, the factors applied by IRUGA make the evaluation of the document independent of any other document in the document set. These factors are the percentage of the query keywords that exist in the document, the minimum term distance (MTD) between word, the location of MTD within the document, and the average HTML tag weigh for the keywords. According to (Green, 2004), Google concentrates mainly on the incoming link to the web page and outgoing link from this page, and applies PageRank concept to rank the retrieved documents accordingly. Therefore, the techniques applied by CSE evaluate the document depending on other documents in the search space. However, it is found that this factor is heavily affected by commercial aspects. As per (Gavin, 2005; Smith, 2007), there are special sites that provide sites with high PageRank, where site owner can include a link to these sites and can request these sites to include a link to his site so that the new site will have high PageRank and consequently this

new site will appear at high position by the CSE.

Additional factors are considered by CSE when retrieving a document. These include: the goal of CSE is to make money, UI is extremely important, real-time/fast expectation, content of web page not sufficient to imply meaning, result ranking cannot assume independence, must consider maliciousness, no quality control on pages (quality varies), web is large (practically infinite), millions of heterogeneous users.

In fact, the technique of this thesis is not proposed to completely replace the search engine of CSE; rather, it is proposed to be combined with the existing technique applied by CSE.

5.8 Summary

This chapter compares IRUGA with the traditional IR approaches that are widely used in web search, highlighting their main drawback, namely, the extensive computation when evaluating the entire document set in response to the user query. In order to cope with those types of approach, a similar model (TPBTIR) is proposed in this chapter which applies the TPEF function to measure the relevance of all the documents set before ranking them and displaying them to the user.

Several experiments are conducted in this chapter to study TPBTIR performance. The first experiment shows the high performance achieved by TPBTIR in terms of the three precision and recall measures. The second experiment compares IRUGA and TPBTIR when both are using the TPEF function in evaluating the documents. There is a level of confidence that, when considering all documents in the space while measuring recall and precision using term-proximity function, the results of TPBTIR are better than IRUGA's. This is because of the probabilistic and random behaviour of IRUGA, which misses some good quality documents while creating the initial population, giving the advantage for TPBTIR to include them all. However, these high measures achieved by TPBTIR are at the expense of the time elapsed before presenting the results to the user, and this leads to the third experiment that examines the time required to produce the results using IRUGA and TPBTIR. Depending on the criteria of selecting the document for evaluation, the performance differs. It is found that the time required by IRUGA is constant since it depends on the population size and the chromosome length regardless of the number of the documents in the search space that

satisfy the selection criteria, while the time consumed by TPBTIR is completely dependent on the number of documents that satisfy these criteria, and if there is a high number of documents having such criteria, this will require a longer time to produce the results. In fact, this is the most important drawback of TPBTIR which makes it impractical for a large set such as the web. The fourth experiment compares the performance of the proposed TPEF against two well known evaluation functions in this arena, namely, the Okapi-BM25 and Bayesian inference network model functions. These comparisons are done in two directions. The first one analyzes the quality of the documents retrieved by comparing the first retrieved document by each function. The results obtained favour the first technique which is TPEF as the documents retrieved are more relevant and can be evaluated independently of other documents in the search space. The second direction of the evaluation is to study the performance in terms of precision and recall measures. Once again the TPEF outperforms these two functions, since the improvement ranges between 37.21% and 84.81% for the precision measure, 16.44% and 18.87% for the recall measure and 5.29% and 10.16% for the P@R measure.

Despite the slightly lower performance of IRUGA as compared to TPBTIR in terms of recall and precision, the performance of IRUGA is still much better when time is concerned where the processing time depends on constant factors, namely, the population size and the chromosome length, whereas in TPBTIR it is completely dependent on the size of the document set, which makes it impractical for web search domain. Moreover, IRUGA's performance is of a high level and is very acceptable when compared with other similar GA-based IR approaches, as illustrated in the previous chapter, where the recall reached 86% at top 100 retrieved document and the precision reaches 86% at top 10 retrieved documents.

Comparing the results achieved with other GA-based IR systems are discussed in the in this chapter as well. IRUGA has been compared with several research works in the IR domain that adopted the GA model. The comparison was done for four measures. The first one is the speed of convergence where the maximum number of generations is examined. IRUGA was one of the fastest convergence techniques since it converges on average after 22.26 generations, while this measure is not applicable for TPBTIR. That means it doesn't fall into local optima. Rather, it converges into global optima with a very reasonable solution. The second measure is Precision at Rank N (P@N) where the common measurement is the 11-

point average precision. The precision achieved for IRUGA using this measure is 0.49 with enhancement of 92.1% over other IR techniques, while the precision of TPBTIR is 0.57 with enhancement of 123.5%. The achieved recall is 0.74 for IRUGA with enhancement of 99.4%, assuming that this measure uses the 11-point average recall. The third measure is Recall at Rank N (R@N). IRUGA achieved 0.74 and TPBTIR achieved 0.85. Finally, the common measure is the precision at recall (P@R) measure. Both IRUGA and TPBTIR achieved very high scores reaching 0.95 and 0.96 respectively. This implies that both models enhanced the precision-recall by at least 35.7 %.

These results meet the aim of this thesis which is to enhance recall and precision for IR systems using GA. Moreover, it puts IRUGA in the top position among similar GA-based IR models, while TPBTIR can be excluded from these results as it has low time performance when applied in the web search domain.

Chapter Six: Conclusions and Future Work

6.1 Introduction

This thesis proposes two IR models; the first one is IRUGA, which is a GA-based IR approach. This approach introduces modified GA operators that allow IRUGA to achieve high performance. The second IR model is TPBTIR, which is based on a traditional IR approach. Both are used to enhance the precision and recall of the web search by means of improving the document representation where an enhanced inverted index is developed for this purpose. Moreover, these two models use the same proposed evaluation function for measuring the document relativity to the user query.

This chapter includes a summary of the IRUGA and TPBTIR design in Section 8.2, followed by listing the contributions of this thesis to the knowledge in Section 8.3. This chapter ends in section 8.4 by outlining some limitations of the proposed models and highlighting future research work that may build on the work described in this thesis.

6.2 Summary: IRUGA and TPBTIR Design and Performance

6.2.1 IRUGA design and performance

IRUGA is composed of two main parts. The first one is the document representation, and the second one is the GA engine which matches the user query with a set of documents relevant to this query and displays the resulted document in descending order.

The document representation model adopted in IRUGA is the *enhanced inverted index* which customizes the known inverted index to match the requirements of IRUGA.

What distinguishes IRUGA from other GA-based IR techniques is the elaboration done on all GA operators and the suggested evaluation function (fitness function). These operators include initial generation creation, parent selection, crossover and mutation, while documents are favoured for selection according to a fitness value obtained by fitness

function. IRUGA is applied to a set of 8344 HTML documents which forms sample of web documents. Moreover, the GA unit of IRUGA is controlled by a set of parameters which are the population size, chromosome length, crossover probability, mutation probability, and termination criteria. According to the empirical study, these parameters are set to be 125 for population size, 125 for chromosome length, 1 for the crossover probability and the mutation rate is 0.7, while the GA unit terminates when the difference in fitness value between two consecutive populations is below threshold, or the maximum number of generations reaches 50.

To decide which technique needs to be adopted for each operator, an extensive empirical study is conducted to analyze the behaviour of existing techniques from the performance and time perspective. However, it is found that some techniques can be further improved to produce better results. The initial selection technique used to implement the initial selection operator is the *selective random selection* technique, while the *binary tournament selection* technique is applied as the parent selection operator. A new technique is developed for the crossover operator which is the *hybrid crossover* technique. The classical random mutation technique is applied in IRUGA.

Moreover, an innovative evaluation function which is adopted here as a fitness function is called *term-proximity evaluation function* (TPEF). It is developed to best reflect the document's relevance to the user query. This fitness function is based on the local factors obtained from the document itself to make the document evaluation independent from other documents in the collection and to speed up the evaluation process of the document. Moreover, term proximity concept is applied in this function which favours documents having query keywords adjacent to each other and having these keywords appearing as close as possible to the beginning of the document. In addition to that, documents having all query keywords are preferred by this function.

Four measures are used to evaluate IRUGA. The first one is the time of convergence. The other three measures are based on the precision (P) and recall (R) concept, where precision is the percentage of relevant retrieved documents to the total retrieved documents, and recall is the percentage of relevant retrieved to the total relevant documents. The first measure of these three is the precision at top N measure ($P@N$). This measure obtains the number of

relevant documents at top N ranked documents. The second one is the recall at top N measure (R@N). This measure obtains the percentage of relevant document to the total relevant at top N ranked documents. The third one is the precision at N recall (P@R) which evaluates the precision for each 10% of the relevant retrieved documents.

IRUGA achieved significant results in these measures for all operators when compared with other GA-based IR systems. For each operator, a comparison is done between several known techniques based on these measures. When comparing the performance of IRUGA with other GA -based IR models, it is found that IRUGA achieved an enhancement of 237.97% for the P@N measure, 136.49 for the recall measure, and 95.71% for the precision-recall measure.

For the convergence speed, which represents the average number of generations required to generate the final results, it is found that this average for IRUGA is 22.3. Compared with that of other GA-based models in literature, it is found that their average ranges between 20 and 500. This raises IRUGA to the top of these approaches where better results are obtained faster without falling into local optima. This implies also that IRUGA is less computationally costly than those approaches.

To demonstrate the beauty of the developed TPEF function which evaluates each document and measures its relevance to the user query, a comparison is done between two known fitness functions in this domain and the proposed TPEF function. The comparison is done by analyzing the first document retrieved by each one of these functions. The results obtained show the high relevance of the top ranked document retrieved by TPEF as compared with that obtained by other models.

6.2.2 TPBTIR design and performance

TPBTIR is another model proposed in this thesis as an IR approach. This model uses the same enhanced inverted indexing model and the same TPEF evaluation function applied in IRUGA. However, what distinguishes this model is that the evaluation is applied on all documents in the set. This is to overcome the probability feature of the GA unit of IRUGA which may miss some high quality documents due to the randomness of selection of the initial population. Therefore, TPBTIR evaluates all the documents in a set and obtains the

best one of them. Therefore, it is found that if the document set is relatively small (less than 20,000 documents), then this approach has very high performance in terms of recall and precision. On the other hand, if the document set is very large, such as the web (which is the aim of this thesis), then the performance of TPBTIR drops dramatically in term of the time required to generate the results, making it inefficient for large-scale space. Nevertheless, its performance in terms of recall and precision measures is slightly better than IRUGA's.

6.3 Contributions

The main contributions of this thesis are the following:

- It develops a customized enhanced inverted index that replaces the common vector space indexing model. The enhanced inverted index facilitates the addition of as much information associated to each indexed word as needed. It encapsulated the word frequency, weight, offset within the sentence and within the document. It is fast to build where the time complexity of building the index using this technique is $O(n)$ compared with $O(n^2)$ used by *vector space* model and $O(n^3)$ required by *latent semantic indexing* model. It consumes very little time to retrieve the needed data which is $O(n)$, and it requires very little storage space compared with the vector space model. It is shown in Chapter 3 that the developed inverted index reduced the required storage space by 99.5%.
- It provides an extensive survey for the existing GA operators that are applied to the IR domain and highlights their strengths and weaknesses. This thesis provided a qualitative comparison between GA operators of the techniques applied in IR domain which researches can benefit from. In addition, this analysis helped in designing an effective combination of operators that helped achieving an enhancement ranging between 20.6% and 237.97% for the precision measure and enhancement ranging between 3.43% and 136.49% for recall measure.
- It develops an innovative evaluation function for evaluating the documents. This function has several features. The first one is the usage of the ratio of the existing query keyword within the document to the total query keywords. In addition, this function is characterized by utilizing the term proximity concept. The third feature of

- this function is the usage of local factors only. In contrast to global factors, evaluating local factors prevents accessing all documents in the space which speeds up the document evaluation process. This function is featured also by having an upper limit. Therefore, the last two characteristics enable the judgment of the document independently of the rest of the documents in the collection. Comparing TPF with OKAPI-BM25, it is found that TPF achieved enhancement in average recall by 27.55% and average precision of 30.58%. Compared with Bayesian, TPF achieved enhancement of 22.52% for recall and 70.43% for the precision.
- It develops a Hybrid crossover operator that has a significant effect on speeding up the convergence process of GA without falling into local optima and it provides high quality chromosomes; in addition, this technique doubles the quality of the produced chromosomes since the best genes of both parents are inherited to the offspring. Compared with the non-ordered crossover, 2-point crossover and crossover technique which produce two offspring, it is found that the hybrid crossover technique achieved maximum enhancement of 237.97% for the precision measure and maximum enhancement of 136.49% for the recall measure.
 - It proposes the TPBTIR system as a traditional IR approach which uses the same indexing unit of IRUGA and the same evaluation function. This model is compared with IRUGA and shows better results in terms of recall and precision. TPBTIR shows to be better than IRUGA by 16.32 % in terms of precision and 6.25% in terms of recall, but its time performance is poor for large collections, which, and favours TPBTIR for a. Since IRUGA is better than other GA-based IR modes and TPBTIR is better than IRUGA, this implies that TPBTIR is better than IRUGA for a small collection (less than 20,000 documents) and IRUGA is better solution for large collections such as web search.

Details of the above contributions are explained in the following subsection.

6.3.1 Effectiveness of the enhanced inverted index

Vector space indexing model forms the majority of the document representation algorithm that is applied in IR systems. The drawbacks of this model are mainly the huge storage space required to store the index, the limitation of adding the required data associated to

each indexed term and the long retrieval time of needed data since scanning the document database is performed sequentially. These drawbacks were behind the selection of a more efficient indexing technique. This thesis suggests the inverted index over other indexing techniques because it is fast to construct, requires small storage space, allows adding as much needed data per indexed word easily without affecting the retrieval process, and retrieve the needed data. According to the experiments conducted in this thesis, it is found that the storage space required by the enhanced inverted index is less than that required by the vector space by almost 99%. Moreover, storing and retrieving the needed data takes a constant time as the targeted data is accessed directly rather than searching for the needed data sequentially.

6.3.2 Potential for improvement of the GA operators

Chapter 3 has analyzed several techniques in literature applying GA operators. This analysis is conducted in order to find the most suitable ones that enable IRUGA to achieve its aims. Starting from the creation of the initial generation, the proposed technique is selective random selection, which selects the documents that have at least one keyword from the query. Such technique of initial selection reduces the domain by almost 66%. This leads to speeding up the convergence and minimizing the processing cost. Moving to the next operator, which is parent selection, it is found that the tournament selection is more suitable for IRUGA as it reduces the time of computing the probability slot for each individual; also it allows better parents to participate in the crossover process. Among 12 crossover techniques analysed in literature review, IRUGA proposed the hybrid crossover which is applied with probability of one. Finally it is found that the most suitable mutation technique is to replace a random gene with a better performing randomly selected gene to maintain the quality of the genes within the chromosome. Applying this technique with probability equal to 0.7 produces slight improvement in precision and recall which reaching to 3.47% and 5.1%. Chapter 3 conducted several experiments to set proper parameter values for the IRUGA's GA unit operators. These parameter values proved their influence on the results obtained when IRUGA produced better than expected results.

6.3.3 Effectiveness of the proposed hybrid crossover technique

In chapter 3, a new crossover operator is presented. This crossover technique is called

hybrid crossover. The concept behind this technique is to collect as many good performance genes as possible in one chromosome, then maintain and push these genes into the next generation. So IRUGA guarantees that the best building block of these genes is not broken or destroyed throughout the crossover operation from generation to generation. This operator played a vital role in speeding up the convergence without falling into local optima. It managed to improve the speed of converges by 25.79% compared to the 2-point crossover technique and by 94.97% compared to the 1-point crossover technique that produces two offspring. However, the non-ordered crossover produced the results much faster (by 38.68%) than the hybrid crossover, but these results are lacking in their efficiency in terms of recall and precision measure since the hybrid crossover enhanced the precision by 61.36% and enhanced the recall by 65.6%.

6.3.4 Effectiveness of improvement of the GA operators

Chapter 4 includes extensive experiments that are applied to the GA operators. The proposed techniques are compared with the existing ones in terms of the three measures, precision at top N (P@N), recall at top N (R@N) and precision at recall (P@R). Each operator is examined using these three measures and the results are presented graphically and numerically. In all cases, the GA unit operators of IRUGA outperform the performance of other existing techniques. Results summarized in table 4-29 show that IRUGA achieved enhancement ranging between 20.60% and 237.97% for P@N is measure excluding the mutation technique, and enhancement ranging between 3.43% and 136.49% for R@N measure and enhancement ranging between 4.6% and 130.07% for P@R measure excluding the mutation technique which achieved very low enhancement.

6.3.5 Efficiency of the term-proximity fitness function

Chapter 3 introduced two evaluation functions which are selected as the fitness function of IRUGA. The first one is the multi-terminal function which used a combination of local and global factors. However, this function does not achieve the expected results. Hence, a second function was developed. This function is called Term-Proximity Fitness Function (TPFF). This function is constructed using local factors only. From the experiments conducted, this function was able to distinguish the relevant documents effectively. This function was compared with two very well known functions in the IR domain, namely,

OKAPI-BM25 and the Bayesian Inference Network model. The results obtained using both IRUGA and TPBTIR approaches show the superiority of TPDF. Details of these improvements are demonstrated in Table 5-8 (in Chapter 5) reflecting enhancement of TPDF using IRUGA by 96% over using Bayesian Inference Network model and by 53.125% using the OKAPI-BM25 in terms of P@N measure. For R@N measure, IRUGA enhanced the results by 12.68% compared to Bayesian inference network model and by 17.65% compared to OKAPI-BM25. In terms of precision-recall measure, it is found that TPDF achieved enhancement of 4.4% compared to the Bayesian model and enhancement of 11.76% compared to the OKAPI model.

6.3.6 Efficiency of IRUGA and TPBTIR

As shown in Chapter 4, IRUGA achieved great enhancement for the three measures (P@N, R@N, and P@R) for all operators when compared with other GA-based IR techniques. For each operator, the comparison was performed between several known techniques based on the mentioned measures. When comparing these techniques for each operator of the GA unit of IRUGA separately, the highest improvement for P@N measure is 237.97%, for R@N measure is 136.49%, and for P@R measure is 130.07%, while when comparing IRUGA with other GA-based and TIR-based approaches, it is found that the improvement is 92.1% for P@N, 99.4% for R@N and 35.7% for P@R. And this is exactly the aim of this thesis: to enhance recall and precision in web search using GA.

Beside IRUGA, Chapter 5 proposed a traditional IR model (TPBTIR) which uses the same indexing unit and the same TPDF evaluation function of IRUGA. However, this model differs from IRUGA in the number of evaluated documents. TPBTIR evaluates the entire document set to obtain the best documents among the set, in contrast to IRUGA which evaluates only the selected documents of the initial population. TPBTIR shows very high results that compete and outperform IRUGA in terms of the three of the measures discussed above. This high performance is restricted by the document set size. The performance of TPBTIR drops dramatically if the collection size is huge, such as in the web. Therefore, this approach is recommended for small collection size that is less than 20,000 documents, while IRUGA is the most suitable GA-based technique to be applied on web search, where its time performance is independent of the collection size.

6.4 Limitations and Future Work

IRUGA is developed to solve the web search problem, which is to retrieve only and all relevant documents in response to the user query. GA is adopted to be one of two main units of IRUGA. In the web, the number of documents that need to be evaluation is huge. As per Google, for a given query there may be more than one million relevant documents. In order for IRUGA to simulate such commercial search engines, it needs to be applied to a very large collection. At least a set of 200,000 documents could be enough to test it, since this is the maximum size used by researchers to examine their techniques (refer to table 5-3). However, such a large set of document could not be obtained for various reasons. Nevertheless, a document set of 8344 was used to examine the proposed technique.

The proposed evaluation function in Chapter 3 assumes that the keywords within the query are unique; i.e. no word is duplicated in the query. In fact, this forms a sort of limitation that needs to be generalized by allowing duplicate keywords to be included in the query.

Searching the web is a wide area which opens up broad prospects for researchers to carry out the development of many techniques and algorithms that aim at improving the quality of the results extracted from the search space. These techniques differ in many factors, such as type of documents, size of document collection, document evaluation techniques and the approach (traditional IR, probabilistic IR, evolutionary, etc). One of these approaches is Genetic Algorithm. It has been adopted for this model due to the outstanding features mentioned in the previous chapter. Despite the potential enhancements done by IRUGA, there are still opportunities for further improvement. Moreover, these improvements consume time required to retrieve the needed documents, the quality of retrieved documents or the recall and precision of the overall retrieved documents.

Recently, XML documents have been introduced in the web. Consequently, IRUGA can be modified to index such documents. The evaluation function can also be adjusted to match such a document type.

Another area of improvement for this research is to increase the document set in order to accurately simulate the web search environment. The large document set will demonstrate the advantage of IRUGA in terms of time performance compared with traditional IR and

TPBTIR. In traditional IR, all documents $D \in S^+$, where S^+ is the total search space, need to be evaluated, ranked and then displayed to the user, while the number of evaluated documents in GA depends on the population size P_s and the chromosome length C_l . When $P_s \times C_l$ is much smaller than S^+ , the advantage of IRUGA appears clearly, and it is the case when applying IRUGA to web search space that has a huge document set.

The evaluation function proposed in this thesis forms a wide area of improvement. One of the factors of this function is the minimum distance between query keywords within the document. In fact, evaluating this factor is the main source of the IRUGA's slowness. In the worst case, calculating this factor requires $O(n \times m)$, where n is the query length and m is document length. This scenario occurs when the query keywords form the whole document making the complexity as high as $O(n^2)$. There is a need to improve the algorithm by calculating the minimum distance in shorter time and thus reduces the retrieval time.

Moreover, there is an open area to control the chromosome length in such a way that it includes the maximum number of relevant documents only. By doing this, the chromosome will have the ultimate P@N score. In this case N will be the chromosome length which results in a score of 1 for each 10% of the chromosome length. Besides, if it has all the relevant documents, then the recall will be 1 for R@N score. By developing such a technique, both measures will have a score of 1. Consequently, P@R measure will also achieve a score of 1. This is because for each 10% of relevant documents retrieved, all the documents within this range are relevant. This is still the ultimate aim of any developed search mechanism.

BIBLIOGRAPHY

- Aickelin, U. (1999). *Genetic algorithms for multiple-choice optimisation problems*. Viewed 16 November 2011, <http://eprints.nottingham.ac.uk/306/1/99thesis.pdf>
- Alam, H., Kumar, A., Nakamura, M., Rahman, F., Tarnikova, Y. and Wilcox, C. (2003). Structured and unstructured document summarization: Design of a commercial summarizer using lexical chains. *In proceedings of the 7th international conference on document analysis and recognition* IEEE. pp. 1147–1152..
- Al-Dallal, A. and Shaker, R. (2009). Genetic algorithm in web search using inverted index representation. *IEEE-GCC Conference and Exhibition, 2009 5th*, pp. 1-5. Kuwait City.
- Alfonseca, M. (1991). Genetic algorithms. *In proceedings of the international conference on APL*, pp. 1-6. ACM Press.
- Aly, A. (2007). Applying genetic algorithm in query improvement problem. *Information Technologies and Knowledge*, vol.1, pp. 309-316.
- Alzahrani, S.M.; Salim, N., (2009), On the use of fuzzy information retrieval for gauging similarity of Arabic documents, Applications of Digital Information and Web Technologies, ICADIWT '09. Second International Conference on the Digital Object, pp.: 539 – 544. IEEE Conference Publications.
- Application development: PL/SQL, Java or C++?* (2002, March). viewed 30 November, 2011, from Search Oracle: <http://searchoracle.techtarget.com/answer/Application-development-PL/SQL-Java-or-C>
- Asllani, A. and Lari, A. (2007). Using genetic algorithm for dynamic and multiple criteria web-site optimizations. *European Journal of Operational Research*, vol. 176, no.3, pp. 1767-1777.
- Ashraf, F., Ozyer, T. and Alhajj, R. (2008), Employing Clustering Techniques for Automatic Information Extraction From HTML Documents, *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 38, HYPERLINK "<http://intl.ieeexplore.ieee.org/v-ezproxy.brunel.ac.uk:2048/xpl/tocresult.jsp?isnumber=4603093>" no. 5 pp. 660 – 673
- Atsumi, M. (1997). Extraction of user's interests from web pages based on genetic algorithm. *IPSJ SIG*, vol. 97, no. 51, pp. 13-18.
- Azcarraga, A., Liu, M.D. and Setiono, R.(2012) Keyword extraction using backpropagation neural networks and rule extraction, *The 2012 International Joint Conference on Neural Networks (IJCNN)*, pp. 1 – 7. IEEE Conference Publications
- Beasley, D., Bull, D. R. and Martin, R. R. (1993). An overview of genetic algorithms: part 1. *Fundamentals University Computing*, vol. 15, no. 2, pp. 58-69.
- Beasley, D., Bull, D. R. and Martin, R. R. (1993). An overview of genetic algorithms: part 2, Research Topics. *University Computing*, vol. 15, no. 4, pp. 170-181.
- Bedi, P. and Chawla, S. (2007). Improving information retrieval precision using query log mining and information scent. *Information Technology Journal*, vol. 6, no. 4, pp. 584-588.

- Bhatia, M. and Khalid, K. (2007). Contextual proximity based term-weighting for improved web information retrieval. *Proceedings of KSEM 2007*, pp. 267-278. Lecture notes of AI-4798, Springer.
- Billhardt, H., Borrajo, D. and Maojo, V. (2002). Using genetic algorithms to find suboptimal retrieval expert combinations. *In Proceedings of SAC*, pp. 657-662.
- Bini, T.A., Lange, A., Sunye, M.S. and Silva, F., Stableness in large join query optimization, *International Symposium on Computer and Information Sciences, 2009. ISCIS 2009. 24th* , pp.639-644, 14-16 Sept. 2009 [online]. Available at: <http://intl.ieeexplore.ieee.org.v-ezproxy.brunel.ac.uk:2048/stamp/stamp.jsp?tp=andarnumber=5291898&isnumber=5291799> [Accessed 24/9/2012]
- Callen, B. (2005). Search engine optimization made easy [Online]. Available at: <http://www.seoelite.com> [Accessed 16/4/2007]
- Carlberger, J., Dalianis, H., Hassel, M. and Knutsson, O. (2001). Improving precision in information retrieval for Swedish using stemming. *Proceedings of NODALIDA '01 - 13th Nordic conference on computational linguistics*. Uppsala, Sweden.
- Carroll, J. and Lee, T, A genetic algorithm for segmentation and information retrieval of SEC regulatory filings, *Proceedings of the 2008 international conference on Digital government research*, Publisher: Digital Government Society of North America
- Carthy, D. C. J., Drummond, A., Dunnion, J. and Sheppard, J. (2003), The use of data mining in the design and implementation of an incident report retrieval system, *in Systems and Information Engineering Design Symposium*, Charlottesville, pp. 13-18.
- Chakraborty, U. K., Deb, K., and Chakraborty, M. (1996). Analysis of selection algorithms: A markov chain approach. *Evolutionary Computation*, vol. 4, no. 2, pp. 133-167.
- Chen, H. (1995). Machine learning for information retrieval: neural networks, symbolic learning, and genetic algorithms. *Journal of the American Society for Information Science*, vol. 46, no. 3, pp. 194-216.
- Chen, H., and Dhar, V. (1991). Cognitive process as basis for intelligent retrieval systems design. *Information Processing and Management* , vol. 27, pp. 405-432.
- Chiararella Y. (2001). Information retrieval and structured documents. *Lectures Notes in Computer Science*, 1980: pp. 291–314.
- Chinneck, J. w. 2006, Heuristic for discrete search: Genetic algorithms and Simulated annealing, *chapter from: Practical Optimization: A Gentle Introduction*, [online]. Available at, [HYPERLINK "http://www.sce.carleton.ca/faculty/chinneck/po/Chapter14.pdf"](http://www.sce.carleton.ca/faculty/chinneck/po/Chapter14.pdf)
<http://www.sce.carleton.ca/faculty/chinneck/po/Chapter14.pdf> [Accessed 22/9/2012]
- Cho, w., and Richards, D. (2004). Improvement of precision and recall for information retrieval in a narrow domain: reuse of concepts by formal concept analysis. *IEEE/WIC/ACM International Conference on Web Intelligence*, pp. 370-376. Beijing, China: WI.
- Collard, P., and Escazut, C. (1995). Genetic operators in a dual genetic algorithm. *Proceedings, Seventh International Conference on Tools with Artificial Intelligence*, pp. 12-19.

- Cooper, J., and Hinde, C. (2003). Improving genetic algorithms' efficiency using intelligent fitness functions. In P. Chung, C. Hinde, and M. Ali (Ed.), *16th International conference on industrial and engineering applications of artificial intelligence and expert systems, IEA/AIE '03*, pp. 636–644. Loughborough, UK: Springer, Berlin.
- Craven, M., DiPasquo, D., Freitag, D., McCallum, A., Mitchell, T., Nigam, K., et al. (1998). To extract symbolic knowledge from the world wide web. *Proceedings of 15th National conference on artificial intelligence (AAAI98)*.
- Crestani F., Lalmas M., Van Rijsbergen C. j. and Campbell I. (1998) "Is this document relevant?...probably": a survey of probabilistic models in information retrieval, *ACM Computing Surveys (CSUR)*, vol. 30 no .4, pp. 528-552.
- Cummins, R., and O'Riordan, C. (2006). Evolving local and global weighting schemes in information retrieval. (Boston, Ed.) *Information retrieval* , vol. 9, no. 3, pp. 311-330.
- Cutler, M., Deng, H., Maniccam, S., and Meng, W. (1999). A new study on using HTML structures to improve retrieval, Tools with Artificial Intelligence. *Proceedings. 11th IEEE International conference on on tools with AI*, pp. 406 - 409.
- Cutler, M., Shih, Y., and Meng, W. (1997). Using the structure of HTML documents to improve retrieval. *The USENIX Symposium on Internet Technologies and Systems*, pp. 241–251. Monterey, California.
- Dashti, F., and Zad, S. A. (2010). Optimizing the data search results in web using genetic algorithm. *International journal of advanced engineering sciences and technologies*, vol. 1, no. 1, pp. 016 – 022.
- Deb, K. (1998). Genetic algorithm in search and optimization: the technique and applications. *Proceedings of international workshop on soft computing and intelligent systems*, pp. 58–87. Calcutta, India.
- Description of LSI*. (2009), [online]. Available at http://en.wikipedia.org/wiki/Latent_semantic_indexing [Accessed 1/12/2009]
- Desjardins, G., Godin, R., and Proulx, R. A. (2005). Genetic algorithm for text mining. *Proceedings of the 6th international conference on data mining, text mining and their business applications*, vol. 35, pp. 133-142.
- Dong, H., Hussain, F. K., and Chang, E. (2008). A survey in traditional information retrieval models. *Second IEEE International conference on digital ecosystems and technologies*, pp. 397 - 402.
- Dong L., and Watters C (2004). Improving efficiency and relevance ranking in information retrieval. In *Proceedings of the 2004 IEEE/WIC/ACM International Conference on Web Intelligence (WI2005)*, pp. 648– 651.
- Drias, H., Khennak, I. and Boukhedra, A. (2009), A hybrid genetic algorithm for large scale information retrieval, *International Conference on Intelligent Computing and Intelligent Systems, ICIS 2009*. vol. 1, pp. 842 - 846, IEEE Conference Publications
- Fan, W., Fox, E., Pathak, P., and Wu, H. (2004). The effects of fitness functions on genetic programming-based ranking discovery for Web search. *Research Articles, Journal of the American Society for Information Science and Technology*, vol. 55, no. 7, pp. 628-636.
- Frid, B., Logounova, L., Michailov, A., Nusinzon, O., and Zeltser, L. (1997). *High precision information retrieval with natural language processing*

- techniques*. [Online]. Available at <http://zeltser.com/info-retrieval/> [Accessed 26/9/2009]
- Latent semantic indexing, (2011). [online]. Available at: http://en.wikipedia.org/wiki/Latent_semantic_indexing [Accessed 1/12/2009]
- Fuhr, N. (1992). Probabilistic models in information retrieval. *Computer Journal*, vol. 35 no. 3, pp. 244-255.
- Fuhr, N., and Buckley, C. (1991). A probabilistic learning approach for document indexing. *ACM Transactions on Information Systems*, pp. 223-248.
- Gancarski, A. L., and Henriques, P. R. (2002). Information Retrieval from structured documents represented by attribute grammars. *International conference on information systems modelling*. Rep. Cheque.
- Gavin, P. (2005), Text Link Ads: The definitive guide to link buying, Text Link Ads Inc.
- Glover, E. (2007). *The real world web search problem: bridging the gap between academic and commercial understanding of issues and methods*. [online], Available at: http://langtech.jrc.ec.europa.eu/mmdss2007/htdocs/Presentations/Docs/MDSS_Glover.pdf [Accessed 26/4/2012]
- Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley.
- Goldberg, D. E., and Deb, K. (1991). A comparative analysis of selection schemes used in genetic algorithms. *Foundations of Genetic Algorithms*, pp. 69–93. San Mateo CA: Morgan Kaufmann.
- Gordon, M. (1988). Probabilistic and genetic algorithms in document retrieval. *Communication of the ACM*, vol. 31, no. 10, pp. 1208-1218.
- Green, J.J. (2004). Google PageRank and related technologies, [online]. Available at: <http://www.lazworld.com/whitepapers/PageRank-Technologies.pdf> [Accessed 22/9/2012]
- Guezouli, L. and Kadache, A. (2012), Information retrieval model based on neural networks using neighbourhood, *International Conference on Information Technology and e-Services (ICITeS)*, pp. 1 – 5. IEEE Conference Publications.
- Hemalatha, M. and Sathya Srinivas, D. (2009). Hybrid neural network model for web document clustering, *Second International Conference on the Applications of Digital Information and Web Technologies, ICADIWT '09*, pp.531 - 538. IEEE Conference Publications.
- Holland, J. H. (1975). *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. Cambridge, Massachusetts, London, England: University of Michigan Press, Ann Arbor, MI.
- Hornig, J. -T., and Yeh, C.-C. (2000). Applying genetic algorithms to query optimization in document retrieval. *Information Process Management*, vol. 36, no. 5, pp. 737–759.
- Húsek, D., Snášel, V., Owais, J., and Krömer, P. (2005). Using genetic algorithms for boolean queries optimization. *Proceeding of the Ninth IASTED International Conference internet and multimedia ststems and applications*, pp. 178-184. Honolulu, Hawaii, USA.

- ixCreateStopWordList*. (2002). Available at: Onix Text Retrieval Toolkit: <http://www.lextek.com/manuals/onix/ixCreateStopWordList.html> [Accessed 1/12/2009]
- Kamps, J. (2004). Improving retrieval effectiveness by reranking documents based on controlled vocabulary. *The 21th European Conference on Information Retrieval*. pp. 283–295. Springer-Verlag Berlin Heidelberg.
- Karthik, M., Marikkannan, M., and Kannan, A. (2008). An intelligent system for semantic information retrieval information from textual web documents. In S. N. Srihari, *Computational forensics: second international workshop, IWCF 2008, Washington, DC, USA, August 7-8*, pp. 135–146, Springer.
- Kazarlis, S. A., Papadakis, S. E., and Theocharis, J. B. (2001). Microgenetic algorithms as generalized hill-climbing operators for GA optimization. *IEEE Transaction on Evolutionary Computation*, vol 5, pp. 204-217.
- Kim, J., and Croft, W. B. (2009). Retrieval experiments using pseudo-desktop collections. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pp. 1297-1306.
- Kim, S., and Zhang, B-T. (2003). Genetic mining of html structures for effective web-document retrieval. *Applied Intelligence*, vol.18, no.3, pp.243-256.
- Kim, S., and Zhang, B.-T. (2000). Web-Document retrieval by genetic learning of importance factors for HTML Tags. In *Proceedings of PRICAI Workshop on Text and Web Mining*, pp. 13-23.
- Klabbankoh, B., and Pinngern, O. (2008). *Applied Genetic Algorithms In Information Retrieval*. Retrieved Aug 22, 2009, from <http://www.ils.unc.edu/~losee/gene1.pdf>
- Kleinberg, J., and Tomkins, A. (1999). Applications of linear algebra in information retrieval and hypertext analysis. In *ACM PODS Conference Proceedings* , pp. 185–193.
- Kobayashi, M., and Takeda, K. (2000). Information retrieval on the Web. *ACM Computer. Surveys*, vol 32, no. 2, pp.144–173.
- Kopec, D. and Marsland, T.A. (2012), Artificial Intelligence: Search Methods, [online]. Available at:
http://spider.sci.brooklyn.cuny.edu/~kopec/Publications/Publications/O_5_AI.pdf
[Accessed 21/9/2012]
- Kofax. (2011). Retrieved 6 2011, 1, from Kofax: <http://www.kofax.com/glossary>
- Kosala, R., and Blockeel, H. (2000). Web mining research: A survey. *SIGKDD Explorations*, vol. 2, no. 1, pp. 1-15.
- Koza, J. R. (1992). Genetic programming: on the programming of computers by means. MA, USA.: MIT Press, Cambridge.
- Kui, F. and Juan, W., (2012), An Optimized Features Extraction Algorithm on VSM, *9th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, 29-31 May, pp. 1471 - 1473.
- Kushchu, I. (2005). Web-based evolutionary and adaptive information retrieval. *IEEE Transactions on Evolutionary Computation*, vol. 9, no.2, pp. 117 - 125.
- Lashkari, A.H., Mahdavi, F. Ghomi, V (2009), V.A Boolean Model in Information Retrieval for Search Engines , *International Conference on Information Management and Engineering*,. *ICIME '09*. pp: 385 – 389, IEEE Conference Publications.

- LEE, W. (2007). *Hierarchical web structure mining*. [online]. Available at: <http://www.ieice.org/~de/DEWS/DEWS2006/doc/2A-v1.pdf> [Accessed 4/4/2007]
- Lewandowski, D. (2005). Web Searching, search engines and information retrieval, *Information Services and Use*, vol. 18, no.3, pp. 137-147
- Lili Yan, L., Chen, H, Ji, W. Lu, Y. and Li, J (2009), Optimal VSM Model and Multi-Object Quantum-Inspired Genetic Algorithm for Web Information Retrieval, *International Symposium on Computer Network and Multimedia Technology, 2009. CNMT 2009*, pp. 1 – 4, IEEE Conference Publications
- Liu, B. (2006). *Web Data Mining*. Springer-Verlag New York, LLC.
- Liu, D.-R., Keyword C.-K. and Wu, M.-Y. (2008), Context-based knowledge support for problem-solving by rule-inference and case-based reasoning, *Machine Learning and Cybernetics, 2008 International Conference on* , 12-15 July, vol.6, pp.3205-3210. 2008
- Lopez-Pujalte, C., Guerrero-Bote, V. P., and de Moya-Anegon, F. (2003). Genetic algorithms in relevance feedback: a second test and new contributions. *Information Processing and Management* , vol. 39, pp. 669–687.
- Lopez-Pujalte, C., Guerrero-Bote, V. P., and de Moya-Anegon, F. (2003). Order-based fitness functions for genetic algorithms applied to relevance feedback. *Journal of the American* , vol. 54, no. 2, pp. 152–160.
- Lops, P., de Gemmis, M., Semeraro, G., Musto, C., and Narducci, F. (2012). Content-based and collaborative techniques for tag recommendation: an empirical evaluation. *Journal of Intelligent Information Systems*, pp. 1-21, Springer Netherlands.
- Losee, R. M. (1996). Learning syntactic rules and tags with genetic algorithms for information retrieval and filtering: an empirical basis for grammatical rules. *Information Processing and Management* , vol. 32, no. 2, pp. 185-197.
- Man, K.F., Tang, K.S., and Kwong, S. (1996), Genetic Algorithms: Concepts and Applications, *IEEE Transactions on Industrial Electronics*, vol. 43, no. 5 pp. 519-534
- Manning, C. D., Raghavan, P., and Schütze, H. (2009). *An introduction to information retrieval*. Cambridge, England: Cambridge University Press.
- Marghny, M. H., and Ali, A. F. (2005). Web mining based on genetic algorithm. *AIML 05 Conference*. Cicc, Cairo, Egypt.
- Marques Periera, R. A., Molinari, A., and Pasi, G. (2005). Contextual weighted representations and indexing models for the retrieval of HTML documents. *Soft Computing* , vol. 9, pp. 481-492.
- Martín-Bautista, M. J., and Vila, M. A. (1998). Applying genetic algorithms to the feature selection problem in information retrieval. In *Lecture Notes On Artificial Intelligence (LNAI)* , pp. 272-281. Springer-Verlag.
- Milutinovic V., Cvetkovic D., and Mirkovic J. (2000). Genetic search based on multiple mutations. *IEEE Computer* , pp. 118-119.
- Minaei-Bidgoli, B., and Punch, W. (2003). Using genetic algorithms for data mining optimization in an educational web-based system. *Genetic and Evolutionary Computation Conference*, pp. 2252–2263. Chicago, USA.
- Mitchell, T. M. (1982). Generalization as search. *Artificial Intelligence* , vol. 18, pp. 203-226.

- Muhlenbein, H., and Schlierkamp-Voosen, D. (1993). Predictive models for the breeder genetic algorithm: I. Continuous parameter optimization. *Evolutionary Computation*, vol. 1, no. 1, pp. 25-49.
- Nirkhi, S., and Hande, K. (2008). Optimization of context disambiguation in web search results . *International Conference on Computer Science and Information Technology, ICCSIT '08.*, pp. 820 - 824.
- Noreault, T., McGill, M., and Koll, M. B. (1980). A performance evaluation of similarity measures, document term weighting schemes and representations in a Boolean environment. *Proceedings of the 3rd annual ACM conference on research and development in information retrieval*, pp. 57-76. Cambridge, England.
- Pandey, H. M., Dixit, A. and Mehrotra, D. (2012), Genetic algorithms: concepts, issues and a case study of grammar induction, September 2012 , *CUBE '12: Proceedings of the CUBE International Information Technology Conference*
- Pathak, P., Gordon, M., and Fan, W. (2000). Effective information retrieval using genetic algorithms based matching functions adaption. *33rd hawaii international conference on science (HICS)*. Hawaii, USA.
- Penev, A., and Wong, R. K. (2010). Structure vs. content in hierarchical corpora. *Information retrieval*, vol. 13, no. 6, pp. 636-656. Springer Science+ Business Media, LLC 2010
- Petridis, V., Kazarlis, S., and Bakirtzis, A. (1998). Varying fitness functions in genetic algorithm constrained optimization: the cutting stock and unit commitment problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B, Cybernetics*, vol. 28, no. 5, pp. 629–640.
- Picarougne, F., Monmarché, N., Oliver, A., and Venturini, G. (2002). Geniminer: Web mining with a genetic based algorithm. *Proceedings of the IADIS International Conference WWW/Internet*, pp. 263–270. Lisbon, Portugal .
- Picarougne, F., Monmarché, N., Oliver, A., and Venturini, G. (2002). Web mining with genetic algorithm. *Eleventh International World Wide Web Conference*. Honolulu, Hawaii.
- Picarougne, F., Monmarché, N., Oliver, A., Venturini, G. (2002). Geniminer: Web mining with a genetic based algorithm. *In: Proceedings of the IADIS International Conference WWW/Internet*, pp. 263–270. Lisbon, Portugal.
- Pohl, S., Zobel, J. and Moffat, A (2010), Extended Boolean retrieval for systematic biomedical reviews, *Proceedings of the Thirty-Third Australasian Conference on Computer Science*, vol. 102, pp. 117-126
- Quinlan, J. (1986, 1993). Induction of decision trees. *Machine Learning* , 1, 81-106.
- Radwan, A. A., abdel Latef, B. A., Ali, A. A., and Sadeq, O. A. (2006). Using genetic algorithm to improve information retrieval systems. *proceedings of world academy of science, engineering and technology*, vol. 17, pp. 6-12.
- Raghavan, V., and Agarwal, B. (1987). Optimal determination of user-oriented clusters: An application for the reproductive plan. *Proceedings of the second international conference on genetic algorithms and their applications*, pp. 241-246. Hillsdale, NJ: Lawrence Erlbaum Associates.
- Rech, J., and Althoff, K.-D. (2004), Artificial intelligence and software engineering - status and future trends. *Special issue on artificial intelligence and software Engineering*, KI vol. 3, pp. 5–11

- Russell, S., Norvig, R., (2010), Artificial intelligence: A modern Approach, Third Edition, Pearson Education , Inc.
- Rylander, B. (2001). Computational complexity and the genetic algorithm. *A dissertation presented in partial fulfillment of the requirements for the degree of doctor of philosophy*. University of Idaho.
- Saini, M. Sharma, D. Gupta, P.K . (2011), Enhancing information retrieval efficiency using semantic-based-combined-similarity-measure. *International Conference on Image Information Processing (ICIIP)*, pp. 1 - 4. IEEE Conference Publications
- Salton, G., and Buckley, C. (1990). Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science*, vol. 41, no. 4, pp. 288-297.
- Salton, G., and Buckley, C. (1988). Term weighting approaches in automatic text retrieval. *Information Processing and Management*, vol. 24, no. 5, pp. 513-523.
- Sehgal, A.K., Das, S., Noto, K., Saier, M.K. and Elkan, C., (2009) Identifying Relevant Data for a Biological Database: Handcrafted Rules versus Machine Learning, *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 8, HYPERLINK "<http://intl.ieeexplore.ieee.org/v- ezproxy.brunel.ac.uk:2048/xpl/tocresult.jsp?isnumber=5730188>" no. 3, pp. 851 - 857
- Simon, P. and Sathya, S.S. (2009), Genetic algorithm for information retrieval, *International Conference on Intelligent Agent & Multi-Agent Systems, IAMA 2009*. pp. 1 – 6, IEEE Conference Publications
- Sivanandam, S. N., and Deepa, S. N. (2008). *Introduction to Genetic Algorithms*. New York: Springer Berlin Heidelberg.
- Smith, J. (2007), Red hot traffic in 10 days, [online], Available at: <http://www.danielherzner.com/ebooks/redhottraffic.pdf>. [Accessed 13/8/2007]
- Snasel, V., Moravec, P., and Pokorny, J. (2005). WordNet ontology based model for web retrieval. *Proceedings of international workshop on challenges in web information retrieval and integration*, pp. 220–225. Washington, D.C.: IEEE Computer Society.
- Song, W., and Park, S. C. (2009). Genetic algorithm for text clustering based on latent semantic indexing. *Computers and Mathematics with Applications* , vol. 57, no.11, pp. 1901-1907.
- Sparck Jones, K., Walker, S., and Robertson, S.E. (2000). A probabilistic model of information retrieval: development and comparative experiments - Part 1. *Information Processing and Management*, vol. 36, no. 6, pp. 779-808.
- Spears, W. M., and De Jong, K. A. (1991). An analysis of multipoint crossover. In G. Rawlins, *Foundations of Genetic Algorithms*, pp. 301–315. CA: Morgan Kaufman.
- Steinbach, M., Karypis, G., and Kumar, V. (2000). A comparison of document clustering techniques. *TextMining Workshop*. KDD.
- Stop Word List 1*. (2002). Retrieved December 1, 2009, from Onix Text Retrieval Toolkit: <http://www.lextek.com/manuals/onix/stopwords1.html>
- T. H. Haveliwala, A. G., Haveliwala, T. h., Gionis, A., Klein, D., and Indy, P. (2002). Evaluating strategies for similarity search on the web. *In Proceedings of the 11th International World Wide Web Conference*, pp. 432–442.
- Tate, D. E., and Smith, A. E. (1995). A genetic approach to the quadratic assignment problem. *Computers and Operations Research*, vol. 22, pp. 73-83.

- The 4 Universities Data Set*. (1998). [online]. Available at : <http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/> [Accessed 12/11/2009]
- Tian, C., Tezuka, T., Oyama, S., Tajima, K., and Tanaka, K. (2006). Improving web retrieval precision based on semantic relationships and proximity of query keywords'. (S. Bressan, J. Kung, and R. Wagner, Eds.) *DEXA 2006. LNCS* , no. 4080, pp. 54–63.
- Uematsu, Y., Inoue, T., Fujioka, F., Kataoka, R., and Ohwada, H. (2008). Proximity scoring using sentence-based inverted index for practical full-text search. *Research and Advanced Technology for Digital Libraries* , no. 5173, pp. 308-319.
- Utgoff, P. E. (1989). Incremental induction of decision trees. *Machine Learning* , vol. 4, pp. 161-186.
- Vrajitoru, D. (1998). Crossover improvement for the genetic algorithm in information retrieval. *Information Processing and Management* , vol. 34, no. 4, pp. 405-415.
- Vrajitoru, D. (1997). Genetic algorithms in information retrieval. *AIDRI97: Learning; From Natural Principles to Artificial Methods*.
- Vrajitoru, D. (2000). Large population or many generations for genetic algorithms ? implications in information retrieval. In F. Crestani, and G. Pasi (Ed.), *Soft Computing in Information Retrieval. Techniques and Applications* (pp. 199-222). Physica-Verlag, Heidelberg.
- Vrajitoru, D. (2007). *Natural Selection and Mating Constraints with Genetic Algorithms*. [online]. Available at: http://www.cs.iusb.edu/~danav/papers/dv_sm05.pdf [Accessed 3/4/2008]
- Wang, Z., and Feng, B. (2005). Optimal genetic query algorithm for information retrieval. In J. C. al, *Parallel and Distributed Processing and Applications, Lecture Notes in Computer Science*, vol. 3358, pp. 888-892. Springer-Verlag Berlin Heidelberg.
- Xu, Q., Shen, H., Dai, Y., Cui, B., and Zhou, X. (2008). Achieving effective multi-term queries for fast DHT information retrieval. In *Lecture Notes in Computer Science* pp. 20-35. Springer Berlin / Heidelberg.
- HYPERLINK ["http://dl.acm.org.v-ezproxy.brunel.ac.uk:2048/author_page.cfm?id=81435598158&coll=DL&dl=ACM&CFID=129606728&CFTOKEN=41966658"](http://dl.acm.org.v-ezproxy.brunel.ac.uk:2048/author_page.cfm?id=81435598158&coll=DL&dl=ACM&CFID=129606728&CFTOKEN=41966658) Xu , Y., HYPERLINK ["http://dl.acm.org.v-ezproxy.brunel.ac.uk:2048/author_page.cfm?id=81435599094&coll=DL&dl=ACM&CFID=129606728&CFTOKEN=41966658"](http://dl.acm.org.v-ezproxy.brunel.ac.uk:2048/author_page.cfm?id=81435599094&coll=DL&dl=ACM&CFID=129606728&CFTOKEN=41966658) Deli , Y. and HYPERLINK ["http://dl.acm.org.v-ezproxy.brunel.ac.uk:2048/author_page.cfm?id=81435604114&coll=DL&dl=ACM&CFID=129606728&CFTOKEN=41966658"](http://dl.acm.org.v-ezproxy.brunel.ac.uk:2048/author_page.cfm?id=81435604114&coll=DL&dl=ACM&CFID=129606728&CFTOKEN=41966658) Yu , L. (2009), Efficient annealing - inspired genetic algorithm for information retrieval from web-document June 2009, *GEC '09: Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation*, Publisher: ACM
- Yang, J., and Korfhage, R. (1993). Query optimization in information retrieval using genetic algorithms. *Proceedings of the fifth International Conference on Genetic Algorithms*, pp. 603-613.

- Yang, J.-J., Korfhage, R., and Rasmussen, E. M. (1992). Query improvement in information retrieval using genetic algorithms - A report on the experiments of the TREC project. *In Proceedings of the first text retrieval conference*, pp. 31-58.
- Yeh, J.-Y., Lin, J.-Y., Ke, H.-R., and Yang, W.-P. (2007). Learning to rank for information retrieval using genetic programming. *In Proceedings of ACM SIGIR 2007 Workshop on Learning to Rank for Information Retrieval (LR4IR '07)*, pp. 41-48. Amsterdam, Netherlands.
- Yoshioka, M. and [HYPERLINK "http://dl.acm.org.v-ezproxy.brunel.ac.uk:2048/author_page.cfm?id=81100124421&coll=DL&dl=ACM&CFID=126621907&CFTOKEN=99656279"](http://dl.acm.org.v-ezproxy.brunel.ac.uk:2048/author_page.cfm?id=81100124421&coll=DL&dl=ACM&CFID=126621907&CFTOKEN=99656279) Haraguchi, M., (2005), On a combination of probabilistic and boolean IR models for WWW document retrieval, *Transactions on Asian Language Information Processing (TALIP)*, vol. 4, no. 3, pp. 340 - 356 Publisher: ACM
- Zaman, A.N.K.; Brown, C.G.(2010), Latent semantic indexing and large dataset: Study of term-weighting schemes, *2010 Fifth International Conference Digital Information Management (ICDIM)*, 5-8 July, pp.1-4
- Zhang, B., Chen, Y., Fan, W., Fox, E. A., Goncalves, M., Cristo, M., et al. (2005). Intelligent gp fusion from multiple sources for text classification. *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, pp. 477–484. New York, NY, USA.
- Zhang, X. (2009). *Effective Search in Online Knowledge Communities: A Genetic Algorithm Approach*. Retrieved from MSc Thesis, Virginia Polytechnic Institute and State, Blacksburg, Virginia, USA.
- Zhang, X., Wei, K., and Meng, X., (2012), A XML query results ranking approach based on probabilistic information retrieval model, *9th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), 2012*, pp. 915 – 919. IEEE Conference Publications

APPENDICES

Appendix A: Term Weight Formulas Used In GA Systems

Table 7-1: description of the terminals used in weighting and fitness functions

Terminal	Description
i	Term i in the document.
j	Document j in space.
w_i	Weight of term i in document j
k_i	Frequency of term i in user query Q .
K	Total number of terms in Q .
f_{ij}	Frequency of term- i in document- j .
F_j	Size of document- j (total number of words in document- j)
t_j	Number of unique terms in document- j
h_{ij}	HTML tag weight of term- i in document- j .
N	Total number of documents in space.
tf_{ji}	Frequency of term i in document j
df_i	Total number of documents having term- i .
T	Total number of all terms in space.
T_i	Total number of term- i in space.
rtf	raw term frequency
l	length of the document vectors
l_{avg}	average length of the document vectors
$k1, b$	tuning parameters
lwt	local weight (within-document weight) –the weight used is simple term frequency
gwt	global weight –the weight used is simple term frequency
$qrtf$	Query row term frequency within a document
Cf	frequency of a term in the collection
H	constant
dim	Number of Dimensions
rad	radius
amp	amplitude
hhl	height half life
ahl	amplitude half life
ck	Coordinates
fd	retrieved document
α, β	coefficients in precision fitness
d_i	minimum frequency component when a term occurs in a document
$ D $	Total number of documents retrieved
$r(d)$	Function returns relevance of document, 1 if relevant, and 0

	otherwise.
A	Parameter which determines the value of factors to be used.

Table 7-2: Term weighting formulas used in GA systems

Formula Number	Term Weighting Function	Reference
1	$w_t = idf_t = \log \left(\frac{N}{df_{ji}} \right)$	(Billhardt et al, 2002)
2	TF-IDF $w_{ji} = tf_{ji} \times idf_{ji} = tf_{ji} \times \log \left(\frac{N}{df_{ji}} \right)$	(Cummins and O’Riordan, 2006; Kim and Zhang, 2003)
3	Okapi $Okapi - tf = \frac{tf}{tf + k_1((1 - b) + b \frac{l}{avg})}$	(Cummins and O’Riordan, 2006)
4	BM25 $BM25 - idf_t = \log \left(\frac{N - df_t + 0.5}{df_t + 0.5} \right)$	(Cummins and O’Riordan, 2006)
5	local and global weighting schemes $w_t(d_i, q) = \sum_{t \in q \cap d} (lw_t \times gw_t \times qrtf)$	(Cummins and O’Riordan, 2006)
6	$w_i = \frac{\frac{cf}{df} \times (\log(tf) + \frac{cf}{df})}{2df + l + tf}$	(Cummins and O’Riordan, 2006)
7	$w_{ij} = \frac{tf_{ij}}{\max_k tf_{ik}} \cdot \frac{\log(N) - \log(df_j)}{\log(N)}$	(Vrajitoru, 2000)
8	Bayesian inference network model: $w_i = (d_t \cdot H + (1 - d_t) \cdot \frac{\log(tf_i + 0.5)}{\log(\max tf_i + 1.0)}) \cdot \frac{\log(\frac{N}{n_i})}{\log N}$	(Kim and Zhang, 2003)
9	2-poisson model $w_i = \frac{tf_i}{K_i + tf_i} \cdot \log \frac{N - n + 0.5}{n + 0.5}$ Where $K = k_1((1 - b) + b \frac{dl}{avdl})$	(Kim and Zhang, 2003)
10	Salton and Buckley: $w_{ij} = \frac{(0.5 + 0.5 \frac{tf_{ij}}{\max tf}) \times \log(\frac{N}{n_i})}{\sqrt{(0.5 + 0.5 \frac{tf_{ij}}{\max tf})^2 \times (\log \frac{N}{n_i})^2}}$	(Radwan et al 2006; Aly, 2007)

<p>11</p>	$ \frac{f(t, d) \cdot \log\left(\frac{C_i}{ C_i } + 0.01\right) \cdot \left[-\sum_{i=1}^M P(C_i) \log P(C_i) + P(t) \sum_{i=1}^M P(C_i t) \log P(C_i t)\right] \cdot \log\left(\frac{ D }{ C_i }\right) \cdot W_t'}{\sqrt{\sum_{i=1}^M \left\{ f(t, d) \cdot \log\left(\frac{C_i}{ C_i } + 0.01\right) \cdot \left[-\sum_{i=1}^M P(C_i) \log P(C_i) + P(t) \sum_{i=1}^M P(C_i t) \log P(C_i t)\right] \cdot \log\left(\frac{ D }{ C_i }\right) \cdot W_t'}^2 \right.} $ <p>$f(t, d)$ is the frequency of feature t occurring in d. D is the total number of documents. $IG(C_i, t)$ is the information gain of t on category C_i. W_t' is the position weight of t in the page.</p>	<p>(Kui and Juan, 2012)</p>
-----------	--	-----------------------------

Appendix B: Fitness Functions Used In GA Systems

Fitness functions used in GA systems to measure the document relativity to the user query are presented in the Table 9-3.

Table 7-3