GENETIC ALGORITHM BASED SEARCH ENGINE

By

Uttam Dhabas- 091451

Ritika Dhingra- 091442

Arpit Khandelwal - 091429

Under the supervision of

Dr.Pradeep Kumar

May – 2013

Submitted in partial fulfillment of the Degree of

Bachelor of Technology

In

INFORMATION TECHNOLOGY

DEPARTMENT OF INFORMATION TECHNOLOGY

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY,

WAKNAGHAT

(I)

Certificate

This is to certify that the synopsis entitles, **"Genetic Algorithm Based Search Engine"**, submitted by **Uttam Dhabas, Ritika Dhingra and Arpit Khandelwal** in partial fulfilment for the award of Degree of Bachelor of Technology in Information and Communication Technology to Jaypee University of Information Technology, Waknaghat, Solan has been carried out under my supervision.

This work has not been submitted partially or fully to any other university or institute for the award of this or any other degree or diploma.

Signature of Supervisor ..... Pardeep Kumar

Name of Supervisor Dr. Pardeep Kumar

Assistant professor

Jaypee University of Information Technology,

Waknaghat, HP

Date 29/5/2013

Certificate

This is to certify that the synopsis entitles, **"Genetic Algorithm Based Search Engi..."** submitted by **Uttam Dhabas, Ritika Dhingra and Arpit Khandelwal** in partial fulfilm... for the award of Degree of Bachelor of Technology in Electronics and Communicatio... engineering to Jaypee University of Information Technology, **Waknaghat, Solan has been carried out under my** supervision.

This work has not been submitted partially or fully to any other university or institute for the award of this or any other degree or diploma.

Signature of Supervisor .....

Name of Supervisor Dr. Pradeep Kumar

Jaypee University of Information Technology,

Waknaghat, HP

Date .....

ACKNOWLEDGMENT

Firstly, we would like to thank God for his grace and our parents for their motivation all through this work.

We express our gratitude and sincere thanks to Dr. Pardeep Kumar, dept. of Computer Science and Information & Communication Technology for providing the opportunity to undertake the project under his able guidance. His directions and unparallel support for fetching solutions from the industry has helped us immensely in realization of the project.

The zeal to accomplish the task of formulating the project could not have been realized without the support and cooperation of the faculty members of the ICT department.

Ritika ..... Uttam ..... Arpit .....

Ritika Dhingra **Uttam Dhabas** **Arpit Khandelwal**

ACKNOWLEDGMENT

Firstly, we would like to thank God for his grace and our parents for their motivation all through this work.

We express our gratitude and sincere thanks to Dr. Pardeep kumar, dept. of Information and Technology for providing the opportunity to undertake the project under his able guidance. His directions and unparallel support for fetching solutions from the industry has helped us immensely in realization of the project.

The zeal to accomplish the task of formulating the project could not have been realized without the support and cooperation of the faculty members of the IT department.

.....	.....	.....
Ritika Dhingra	**Uttam Dhabas**	**Arpit Khandelwal**

CONTENT

ABSTRACT

Our project aims at formulation of a search engine which deals with the retrieval of relevant information from the document set based on user query.

The implementation of the project is based on the principle of genetic algorithm.

We picture our project in the two following phases:-

Phase 1-

- we have formulated a prototype search engine habituating techniques such as indexing, database connectivity and page ranking.
- The assorted functions it performs include searching user query, clearing of tables in database and providing options of depth range, re-indexing and full indexing.

Phase 2-

- Implementation of genetic algorithm to enhance the performance of the search engine in ways like relevant data extraction, speed, search optimization, reduction in search time, impaired keyword detection etc.

Events to deal with-

- Speed
- More intuitive interface
- An ability to set preferences
- More, higher-quality search features

CHAPTER - 1

INTRODUCTION

1.1 GA

A GA is an adaptive heuristic search algorithm premised on the evolutionary ideas of natural selection and genetics [33]. The GA approach has gained importance and popularity, as evident in the number of studies that have used it to improve different optimization procedures to be able to find a global solution in many problems. GA have been used for difficult problems , for machine learning and also for evolving simple programs. In this paper; and for each similarity measure in the vector space model we will implement and compare ten different genetic algorithms settings (different mutation techniques, different fitness functions, different crossover techniques) to optimize the user query.

The basic concept of GA is designed to simulate processes in natural systems necessary for evolution. As such they represent an intelligent exploitation of a random search within a defined search space to solve a problem. GAs exploits the idea of the survival of the fittest and an interbreeding population to create a novel and innovative search strategy. A population of strings, representing solutions to a specified problem, is maintained by the GA. The GA then iteratively creates new populations from the old by ranking the strings and interbreeding the fittest to create new strings, which are hopefully closer to the optimum solution to the problem at hand. So in each generation, the GA creates a set of strings from the bits and pieces of the previous strings. The idea of survival of the fittest is of great importance to genetic algorithms.

GAs use what is termed as a fitness function in order to select the fittest string that will be used to create new, and conceivably better, populations of strings. The only thing that the fitness function must do is to rank the strings in some way by producing the fitness value. These values are then used to select the fittest strings.

1.2 BASIC PRINCIPLES

GA's are characterized by 5 basic components as follow:

1) Chromosome representation for the feasible solutions to the optimization problem.

2) Initial population of the feasible solutions.

3) A fitness function that evaluates each solution.

4) Genetic operators that generate a new population from the existing population.

5) Control parameters such as population size, probability of genetic operators, number of generation etc.

CHAPTER 2

PHASE 1- INFORMATION RETRIEVAL SYSTEM

2.1 Information Retrieval (IR)

Information retrieval may be defined, in general, as the problem of selection of document information from storage in response to search questions provided by a user. Information retrieval system (IRS) deals with document databases containing textual, pictorial or vocal information and process user queries trying to allow the user to access relevant information in an appropriate time interval.

An Information Retrieval System consists of a software program that facilitates a user in finding the information the user needs. IR is to provide the users with the documents that satisfy their information need. IRS have to extract the key words from the documents and assign weights for each keyword. The conditions that an IR system should follow to be effective are given as follows: The IRS should be scalable enough to be able to handle large document collections .It must be able to build the indexes in a reasonable amount of time to ensure the index efficiency. Query efficiency must be ensured to find out whether the queries are running fast. Query Effectiveness also affects the IRS since the retrieved result set must be relevant. Research in IR includes modeling, document classification and categorization, systems architecture, user interfaces, data visualization, filtering, languages, etc

The crawler which is a part of information retrieval system is used to gather the information from the websites; here it acts as a document crawler from which we derive the information content. An important aspect that is to be considered about crawlers is the nature of the crawl task. Crawl characteristics such as queries and/or keywords provided as input criteria to the crawler, user-profiles, and desired properties of the pages to be fetched can lead to significant differences in crawler design and implementation. The task could be constrained by parameters like the maximum number of pages to be fetched (long crawls vs. short crawls) or the available memory. A crawling task can be viewed as a constrained multi-objective search problem. Genetic algorithms [5] are good at effectively solving large search and optimization problems. They are a group of stochastic search algorithms discovered in 1960s inspired from evolutionary biology. GA searches a space defined by the representation of the application by iterating the three basic step like Fitness Evaluation, Selection and applying genetic operators.

2.2 Relevance of Genetic Algorithm in Information Retrieval

There are three main components that have to be taken care while designing GA. The first one is coding the problem solutions, next is to find a fitness function that can optimize the performance and finally, the set of parameters including the population size, population structure and genetic operators. Genetic algorithm is a powerful search mechanism and it is suitable for the information retrieval for the following reasons .

The document search space represents a high dimensional space. GAs are one of the powerful searching mechanism known for its robustness and quick search capabilities. So they are suitable for information retrieval. In comparison with the classical information retrieval models, GA manipulates a population of queries rather than a single query. Each query may retrieve a subset of relevant documents that can be merged. GA is more efficient than using a hill climbing algorithm. The traditional methods of query expansion manipulate each term independent of other. GA contributes to maintain useful information links representing a set of terms indexing the relevant documents.

The traditional methods of relevance feed back are not efficient when no relevant documents are retrieved with the initial query. The probabilistic exploration induced by GA permits the exploration of new areas in the document space.

2.3 Functions of the Document Crawler

Document Crawler is used especially for searching information such as journal, conference publications from the online databases and in other document collections. User can give reference to the location where all the documents are stored. The Crawler scans each and every documents and it stores the title of the document, document id, the abstract of the document and the keywords given in the document. It extracts all the information content from the document.

2.4 IR Paradigms

This section briefly describes various research paradigms prevalent in IR and where our work fits in. At a broad level, research in IR can be categorized [Chen, 1995] into three categories:

1. Probabilistic IR
2. Knowledge based IR
3. IR based on machine learning techniques.

2.4.1 *Probabilistic IR:*

Probabilistic retrieval is based on estimating a probability of relevance of a document to the user for the 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 et al., 1991; Gordon, 1988]. There are three different learning strategies used in probabilistic retrieval. Estimation of probabilities of relevance is done for a set of sample documents [Robertson & Sparck Jones, 1976], or a set of sample queries [Maron & Kuhns, 1960] and extended to all the documents or queries. Inference networks [Turtle & Croft, 1990] use a document and query network that capture probabilistic dependencies among the nodes in the network.

2.4.2 *Knowledge based IR*:

This approach focuses on modeling 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 modeled 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 et al., 1991].

2.4.3 Learning systems based IR:

This approach is based on algorithmic extraction of knowledge or identifying patterns in the data. There are three broad areas within this approach: Symbolic learning, Neural networks, and Evolution based algorithms. In the symbolic learning approach knowledge discovery is done typically by inductive learning by creating a hierarchical arrangement of concepts and producing IF-THEN type production rules. ID3 decision-making algorithm [Quinlan, 1986] is one such popular algorithm.

2.5 PROCESS OF OUR SYSTEM

The database for online Web search is an index built from the above repository of collected web documents to create a repository in the lexicon form. The formed lexicon structure associates an ID to every term in the repository. The *termID identifies a list of documents in which the term* occurs, and some contextual information about it, such as position and various other attributes (e.g., is the term in the document title?). The size of the resultant inverted index depends on the specific implementation, but it tends to be on the same order of magnitude as the original repository. The typical search query consists of a sequence of terms, and the system's task is to find the documents that contain all of the terms (an AND query) and decide which of those documents are most likely to satisfy the user. The user query is first received by a front-end Web server and distributed to all of the machines in the index cluster (containing the index). Index-serving machines compute local results, prerank them, and send their best results to the front-end system (or some intermediate server), which selects the best results from across the whole cluster.

The GA algorithm flowchart is illustrated in Figure 1. Genetic algorithm operations can be used to generate new and better generations. As shown in

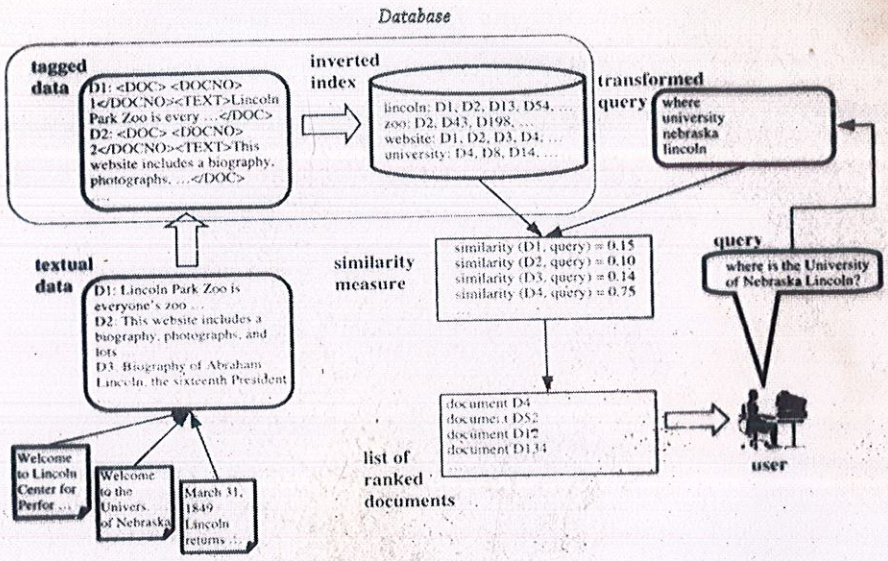


Figure 2.1 : The genetic algorithm operations

2.5.1 Reproduction

The selection of the fittest individuals based on the fitness function.

2.5.2 Crossover

The exchange of genes between two individual chromosomes that are reproducing. In one point cross over a chunk of connected genes will be swapped between two chromosomes. There are many crossover strategies such as n-point crossover , restricted crossover , uniform crossover , fusion operator and dissociated crossover. For more details about the crossover strategies you can see the related references.

2.5.3 Mutation:

Is the process of randomly altering the genes in a particular chromosome. There are two types of mutation:

1. Point
2. Mutation: in which a single gene is changed.
3. Chromosomal mutation: where some number of genes is changed completely.

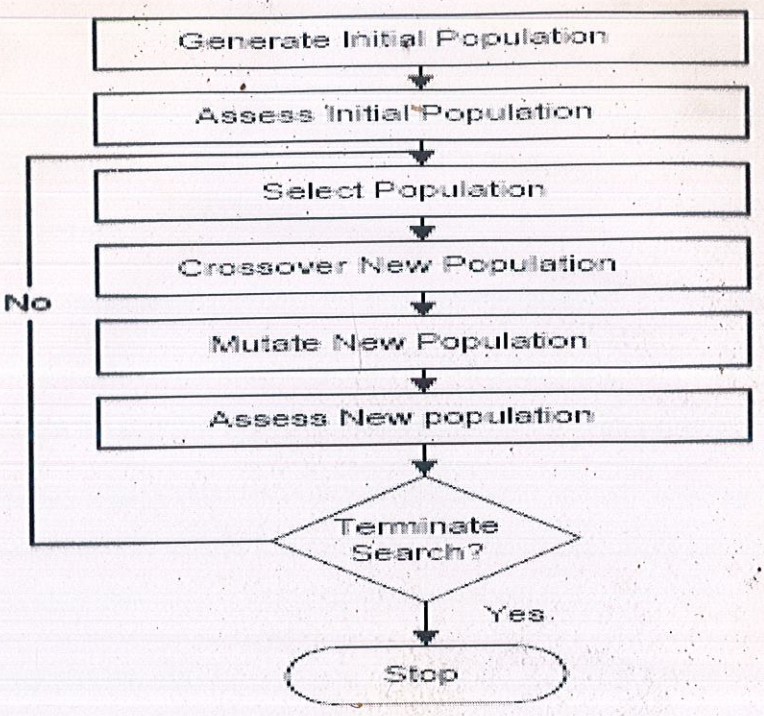


Figure. 2.2: Flowchart for Typical Genetic Algorithm.

2.5.4 Fitness Function:

Fitness function is a performance measure or reward **function which evaluates how each solution** is good.

2.5.5 Selection

Chromosomes selection depends on the fitness function where the higher values have a higher probability to be selected in the next generation.

Introduction

In evolutionary algorithms, the selection operation

Mate = select(Pop, v,ms) chooses ms individuals according to their fitness values v from the population Pop and places them into the mating pool Mate .

On the mating pool, the reproduction operations will subsequently be applied. Selection may behave in a deterministic or in a randomized manner, depending on the algorithm chosen and its application-dependant implementation.

Furthermore, elitist evolutionary algorithms may incorporate an archive Arc in the selection process. Generally, there are two classes of selection algorithms: such with replacement (annotated with a subscript r) and such without replacement (annotated with a subscript w).

In a selection algorithm without replacement, each individual from the population Pop is taken into consideration for reproduction at most once and therefore also will occur in the mating pool Mate one time at most. The mating pool returned by algorithms with replacement can contain the same individual multiple times. Like in nature, one individual may thus have multiple offspring. Normally, selection algorithms are used in variant with replacement. One of the reasons therefore is the number of elements to be placed into the mating pool (corresponding to the parameter ms).

The selection algorithms have major impact on the performance of evolutionary algorithms. Their behavior has thus been subject to several detailed studies, conducted by, for instance, Goldberg and Deb ,Blickle and Thiele , and Zhong et al, just to name a few.

Usually, fitness assignment processes are carried out before selection and the selection algorithms base their decisions solely on the fitness v of the individuals. thus saving the costs of the fitness assignment process.

Many selection algorithms only work with scalar fitness and thus need to rely on a fitness assignment process in multi-objective optimization. Selection algorithms can be chained the resulting mating pool of the first selection may then be used as input for the next one, maybe even with a secondary fitness assignment process in between. In some applications, an environmental selection that reduces the number of individuals is performed first and then a mating selection follows which extracts the individuals which should be used for reproduction.

Truncation Selection

Truncation selection, also called deterministic selection or threshold selection, returns the k < ms best elements from the list Pop. These elements are copied as often as needed until the mating pool size ms reached. For k, normally values like len(Pop)/2 or len(Pop)/3 are

Truncation selection is usually used in Evolution Strategies with $(\mu+\lambda)$ and (μ, λ) strategies. In general evolutionary algorithms, it should be combined with a fitness assignment process that incorporates diversity information in order to prevent premature convergence.

Recently, L¨assig et al. have proved that truncation selection is the optimal selection strategy for crossover, provided that the right value of k is used. In practical applications, this value is normally not known.

In the figure below, we sketch the expected number of offspring for the individuals. In this selection scheme, the diagram will look exactly the same regardless whether we use fitness configuration 1 or 2, since it is solely based on the order of individuals and not on the numerical relation of their fitness. If we set k = ms = len(Pop), each individual will have one offspring in average.

If k = 1 2ms, the top-50% individuals will have two offspring and the others none. For k = 110ms, only the best 100 from the 1000 solution candidates will reach the mating pool but reproduce 10 times in average.

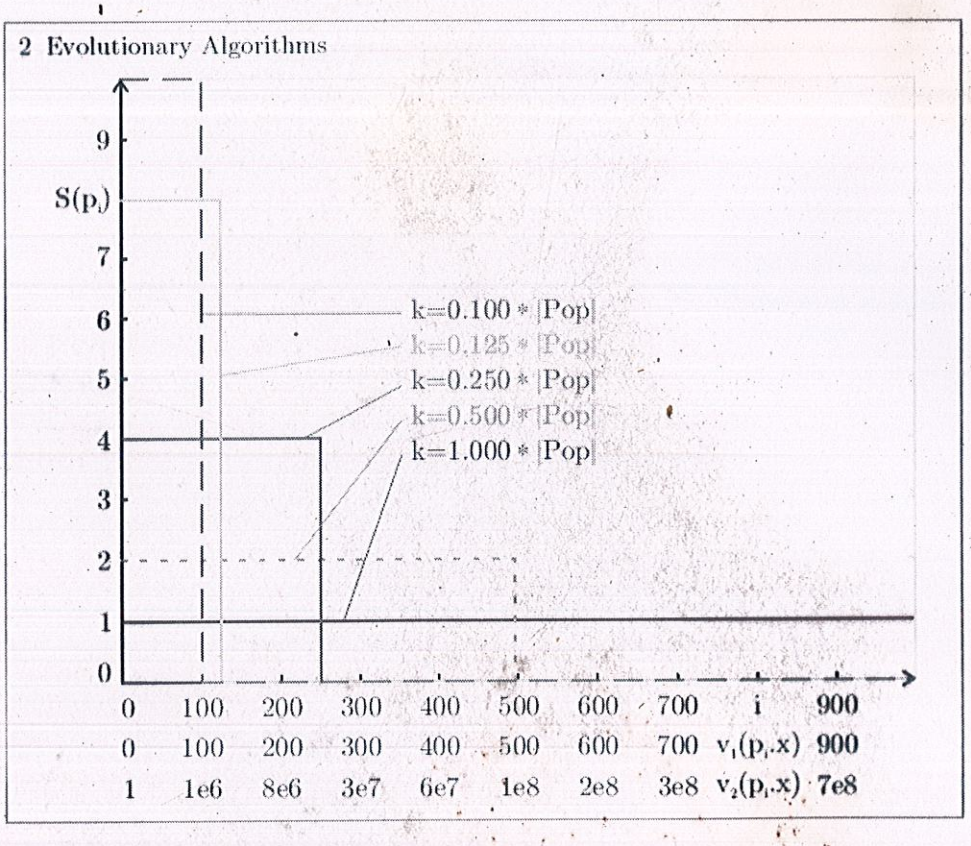


Figure 2.3: Evolutionary Algorithm

Fitness Proportionate Selection

Fitness proportionate selection22 has already been applied in the original genetic algorithms as introduced by Holland and therefore is one of the oldest selection schemes. In fitness proportionate selection, the probability P(p1) of an individual p1 ∈ Pop to enter the mating pool is proportional to its fitness v(p.x) (subject to maximization) compared to the sum of the fitness of all individuals.

There exists a variety of approaches which realize such probability distributions, like stochastic remainder selection and stochastic universal selection.

2.6 Operators:

In our GA approaches, we use two GA operators to produce offspring chromosomes, which are:

2.6.1 Crossover:

it is the genetic operator that mixes two chromosomes together to form new offspring. Chromosomes are not subjected to crossover remain unmodified. Higher fitness chromosome has an opportunity to be selected more than lower ones, so good solution always survives to the next generation.

a)One-point crossover operator.

b)Restricted crossover operator.

c)Uniform crossover operator.

d)Fusion operator.

e)Dissociated crossover.

2.6.2 Mutation

which involves the modification of the gene values of a solution with some probability Pm. chromosome modification using mutation may lead to better or poorer chromosomes. If they are poorer than old chromosome they are eliminated in selection step. In this experiment we used a mutation probability (Pm=0.7) and two different mutation strategies:

a) Point mutation.

b) Chromosomal mutation.

CHAPTER 4
INDEXES

- Indexes are data structures designed to make search faster
- Text search has unique requirements, which leads to unique data structures
- Most common data structure is *inverted index*
  - general name for a class of structures
  - "inverted" because documents are associated with words, rather than words with documents
    - similar to a *concordance*

4.1 INDEXES AND RANKING

- Indexes are designed to support *search*
  - faster response time, supports updates
- Text search engines use a particular form of search: *ranking*
  - documents are retrieved in sorted order according to a score computing using the document representation, the query, and a *ranking algorithm*
- What is a reasonable abstract model for ranking?
  - enables discussion of indexes without details of retrieval model

4.2 INVERTED INDEX

- Each index term is associated with an *inverted list*
  - Contains lists of documents, or lists of word occurrences in documents, and other information
  - Each entry is called a *posting*
  - The part of the posting that refers to a specific document or location is called a *pointer*
  - Each document in the collection is given a unique number

Lists are usually *document-ordered* (sorted by document number)

CHAPTER 4

QUERRY PROCESS

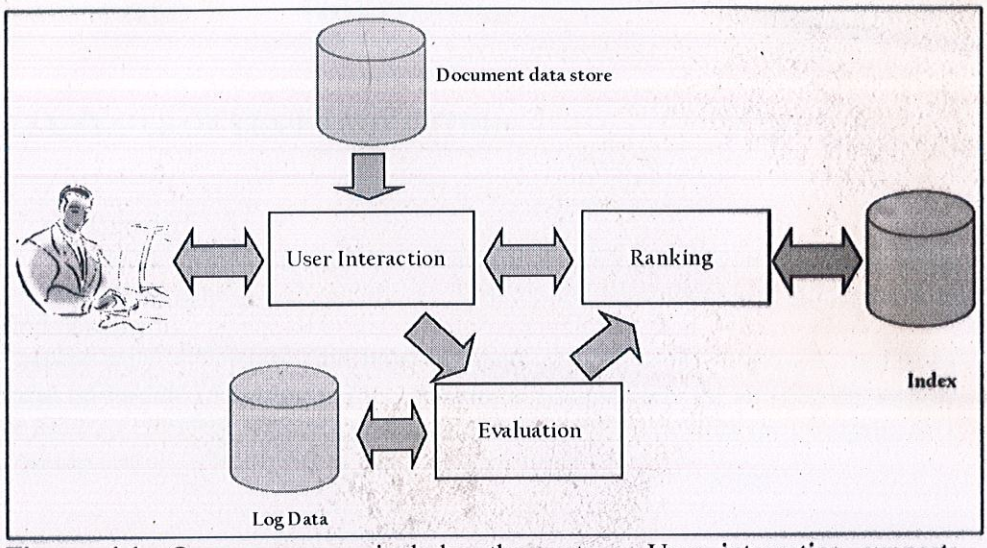


Figure 4.1: Querry process includes three steps. User interaction supports creation and refinement of query, display of results; Ranking uses query and indexes to generate ranked list of documents and Evaluation monitors and measures effectiveness and efficiency (primarily offline).

4.1 INDEX CREATION

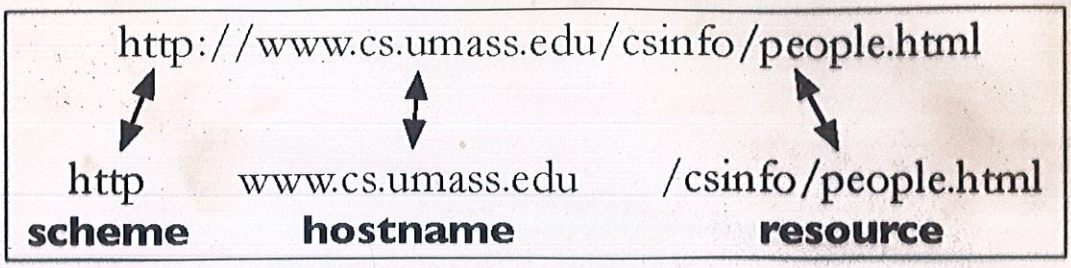
i. Document Statistics
   - Gathers counts and positions of words and other features
   - Used in ranking algorithm

ii. Depth
   - Computes depth for index terms
   - Used in ranking algorithm

Depth can be varied

4.2 RETRIEVING WEB PAGES

Every page has a unique *uniform resource locator* (URL)
Web pages are stored on web servers that use HTTP to exchange information with client software
e.g.,



4.3 DETAILS OF TEXT ACQUISITION

- Crawler

Identifies and acquires documents for search engine. There are many types of crawlers namely web, enterprise, desktop etc. Web crawlers follow *links* to find documents .

Crawler must efficiently find huge numbers of web pages (*coverage*) and keep them up-to-date (*freshness*). There are single site crawlers for *site search, topical* or *focused* crawlers for vertical search and *Document* crawlers for enterprise and desktop search to follow links and scan directories.

4.4 TEXT TRANSFORMATION

- Parser
  - Processing the sequence of text *tokens* in the document to recognize structural elements
    - e.g., titles, links, headings, etc.
  - *Tokenizer* recognizes "words" in the text
    - must consider issues like capitalization, hyphens, apostrophes, non-alpha characters, separators
  - *Markup languages* such as HTML, XML often used to specify structure
    - *Tags* used to specify document *elements*
      - E.g., <h2> Overview </h2>
    - Document parser uses *syntax* of markup language (or other formatting) to identify structure

- Stopping
  - Remove common words
    - e.g., "and", "or", "the", "in"
  - Some impact on efficiency and effectiveness
  - Can be a problem for some queries
- Stemming

- Group words derived from a common *stem*
  - e.g., "computer", "computers", "computing", "compute"
- Usually effective, but not for all queries
- Benefits vary for different languages

- Link Analysis
  - Makes use of *links* and *anchor text* in web pages
  - Link analysis identifies *popularity* and *community* information
    - e.g., PageRank
  - Anchor text can significantly enhance the representation of pages pointed to by links
  - Significant impact on web search
    - Less importance in other applications
- Information Extraction
  - Identify classes of index terms that are important for some applications
  - e.g., *named entity recognizers* identify classes such as *people*, *locations*, *companies*, *dates,* etc.
- Classifier
  - Identifies class-related metadata for documents
    - i.e., assigns labels to documents
    - e.g., topics, reading levels, sentiment, genre
  - Use depends on application

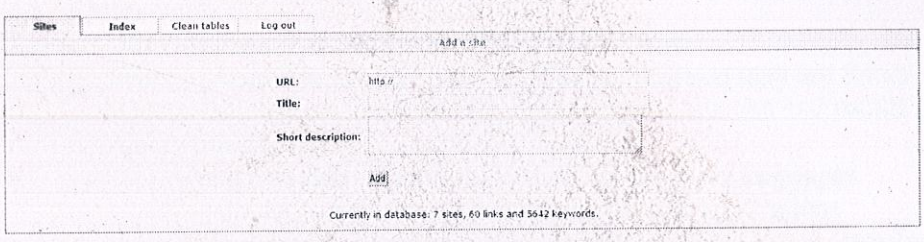
4.5 INDEX CREATION

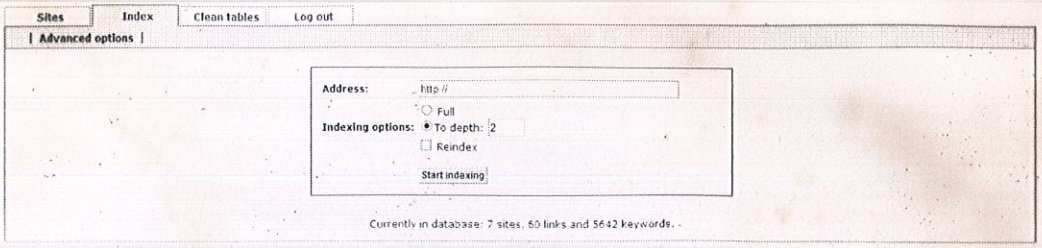
- Document Statistics
  - Gathers counts and positions of words and other features
  - Used in ranking algorithm
- Weighting
  - Computes weights for index terms
  - Used in ranking algorithm
  - e.g., *tf.idf* weight
    - Combination of *term frequency* in document and *inverse document frequency* in the collection

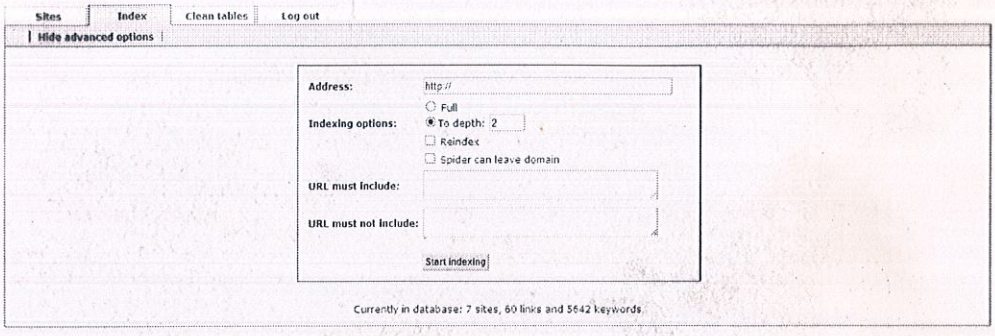
- Inversion
  - Core of indexing process
  - Converts document-term information to term-document for indexing
    - Difficult for very large numbers of documents
  - Format of inverted file is designed for fast query processing

- Must also handle updates
- Compression used for efficiency

- Index Distribution
  - Distributes indexes across multiple computers **and/or multiple sites**
  - Essential for fast query processing with **large numbers of** documents
  - Many variations
    - Document distribution, term distribution, **replication**
  - *P2P* and *distributed IR* involve search across **multiple sites**







4.6 USER INTERACTION

- Query input

  Provides interface and parser for *query language*

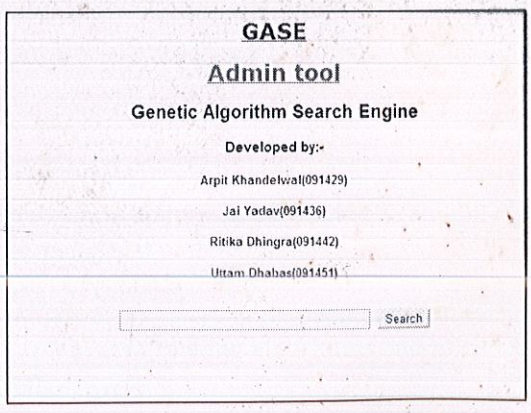
  Most web queries are very simple, other applications may use forms

  Query language used to describe more complex queries and results of query transformation

  e.g., Boolean queries, Indri and Galago query languages

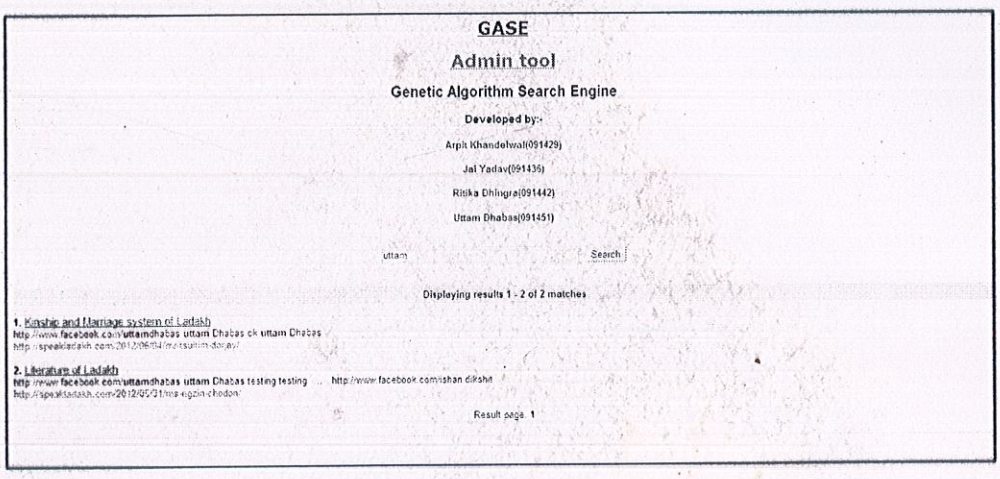
  similar to SQL language used in database applications

  IR query languages also allow content and structure specifications, but focus on content



- Query transformation
  - Improves initial query, both before and after initial search
  - Includes text transformation techniques used for documents
  - *Spell checking* and *query suggestion* provide alternatives to original query
  - *Query expansion* and *relevance feedback* modify the original query with additional terms

- Results output
  - Constructs the display of ranked documents for a query
  - Generates *snippets* to show how queries match documents
  - *Highlights* important words and passages
  - Retrieves appropriate *advertising* in many applications
  - May provide *clustering* and other visualization tools



RANKING

- Scoring
  - Calculates scores for documents using a ranking algorithm
  - Core component of search engine
  - Basic form of score is $\sum q_i d_i$
    - q_i and d_i are query and document term weights for term i
  - Many variations of ranking algorithms and retrieval models
- Performance optimization

- Designing ranking algorithms for efficient processing
  - *Term-at-a time* vs. *document-at-a-time* processing
  - *Safe* vs. *unsafe* optimizations
- Distribution
  - Processing queries in a distributed environment
  - *Query broker* distributes queries and assembles results

Caching is a form of distributed searching

EVALUATION

- Logging
  - Logging user queries and interaction is crucial for improving search effectiveness and efficiency
  - *Query logs* and *clickthrough data* used for query suggestion, spell checking, query caching, ranking, advertising search, and other components
- Ranking analysis
  - Measuring and tuning ranking effectiveness
- Performance analysis
  - Measuring and tuning system efficiency

CHAPTER 5

PHASE 2- IMPLEMENTATION OF GENETIC ALGORITHM

5.1 Usage of Genetic Algorithm in Information Retrieval

There are three main components that have to be taken care while designing GA . The first one is coding the problem solutions, next is to find a fitness function that can optimize the performance and finally, the set of parameters including the population size, genetic operators and their percentages. Genetic algorithms are a powerful search mechanism and it is suitable for the information retrieval for the following reasons. The document search space represents a high dimensional space. GAs are one of the powerful searching mechanism known for its robustness and quick search capabilities. So they are suitable for information retrieval. In comparison with the classical information retrieval models, GA manipulates a population of queries rather than a single query. Each query may retrieve a subset of relevant documents that can be merged. The terms occur in documents as groups. GA contributes to maintain useful information links representing a set of terms indexing the relevant documents.

CLASSES OF SEARCH TECHNIQUES

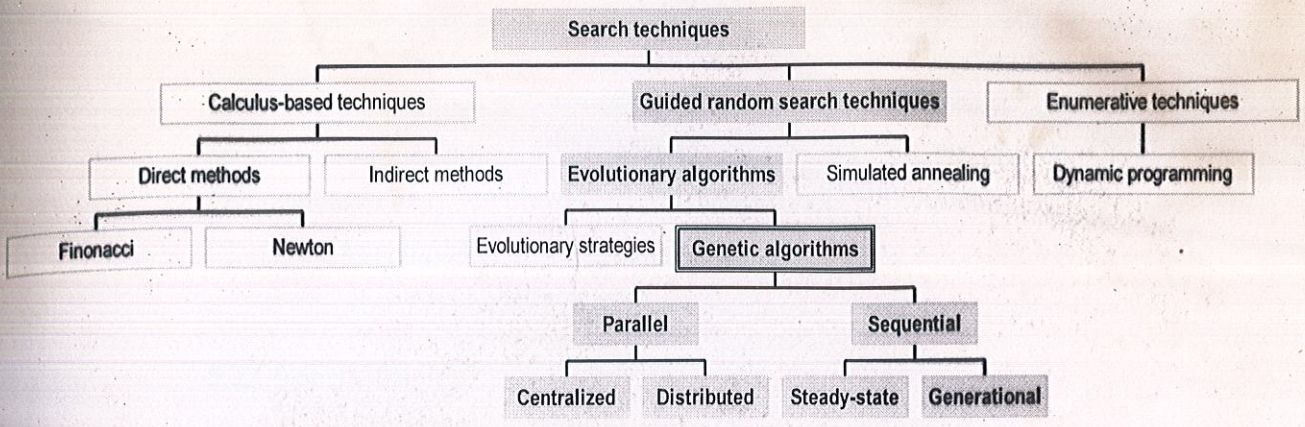


Figure 5.1 : Classes of Search Techniques

5.2 Genetic Model For Information Retrieval

Information Retrieval (IR) may be defined, in general, as the problem of selection of document information from storage in response to search questions provided by a user. Information retrieval system (IRS) deals with document databases containing textual, pictorial or vocal information and process user queries trying to allow the user to access relevant information in an appropriate time interval.

This information retrieval approach given in Fig 2 can be applied for retrieving information. Query and documents are represented as chromosome. An initial population of query is created. The query is sent to the information retrieval system, a match is done between the query chromosome and the document chromosome, and then the document is considered as relevant. If non relevant documents are found, then the query is reformulated. The query is reformulated until a relevant document is retrieved. The document collection consists of many documents containing information about various subject or topics of interests.

The primary concern in representation is how to select proper index terms. Representation proceeds by extracting the key words that are considered as content identifiers and organizing them into the required format.

GAs have been applied to solve some ofthe information retrieval problems. The problem areas include Genetic Mining, Agents for internet search, Query Formulation, Query Optimization, Document Indexing, Ranking, Document Clustering, Match function Adaptation, Rough Sets.

5.2.1 Genetic Mining

Genetic mining is the mining of data by genetic algorithms to improve the efficiency of the document retrieval. Web documents have a number of tags indicating the structure of texts. *Sun Kim* proposed a genetic algorithm that learns the internal structure of HTML documents which are used to re rank the documents retrieved by standard weighing schemes for improving the performance. The use of document structure mining approach tends to move relevant documents to upper ranks, which is necessary in web information retrieval systems. *Ardil* proposed another genetic algorithm in concept weighting and topic identification, based on the concept of standard deviation The term "discourse topic" is used as a representation of the document content in order to distinguish it from other documents in corpus. Discourse topic is an understandable topic which allows a person to quickly see what the topic is about. Concept is used to state some related words that point to a specific entity or impression in a document.

a. Agents for Internet Search

Different proposals are put forward by the authors that use GAs in Internet search to make the search process easier. In , an intelligent personal spider approach for Internet

searching is proposed. Chen et al. implemented Internet personal spiders based on best first search and GA techniques. The used GA applies stochastic selection based on Jaccard's fitness, with heuristic-based crossover and mutation operators. These personal spiders dynamically take a set of user's selected starting homepages and search for the most closely related homepages in the web, based on the existing links and keyword indexing.

b. Query Optimization

Query optimization is defined as the procedure or procedures used to make an information retrieval system effective by improving the query using mathematical techniques. A query with Boolean and logical operators was used in information retrieval. For genetic algorithms, encoding chromosomes was done from Boolean query, where it was represented in the form of tree prefix with indexing for all terms and all Boolean logical operators. Information retrieval effectiveness measures precision and recall were used as a fitness function in the work. Other Genetic algorithms operators used were as single point crossover on Boolean logical operators, and mutation operator was used to exchange one of the Boolean operators AND, OR and XOR with any other one. Roci'o L. Cecchini , Carlos M. Lorenzetti , Ana G. Maguitman, Nelida Beatnz Brignole proposed optimization techniques based on Genetic Algorithms to evolve "good query terms" in the context of a given topic .These techniques place emphasis on searching for novel material that is related to the search context.

The use of a mutation pool to allow the generation of queries with new terms, study the effect of different mutation rates on the exploration of queryspace is discussed and the use of a specially developed fitness function that favors the construction of queries containing novel but related terms. This proposal is a novel GA approach to Web search based on thematic contexts. Proposed method is fully automatic and does not require relevance feedback from

the users.

c. Document Indexing

Indexing is the process which transforms an unstructured set of tokens, such as words or terms, to a data structure, called index. An index is a representation of the content of a collection of documents and of each document. It is by means of indexing that tokens that are found in documents individually or in groups become key words, index terms, or descriptors, thereby assuming the representational power that is needed in order to identify relevant documents. Given an input query, an IR system accesses the indexes and selects the documents probably relevant to the end user's information need represented by that query. In the field of document indexing, the early works were done by Gordon. In his work, he proposed to associate more than a single description to each document and to adapt them throughout time as a good solution to the problem of the different forms that different users' queries searching for the same documents can present. Gordon proposed a GA to derive the document descriptions. He chose a binary coding scheme where each description is a fixed length, binary vector.

The genetic population is composed of different descriptions for the same document. The fitness function is based on calculating the similarity between the current document description and each of the queries by means of the Jaccard's index, and then computing the average adaptation values of the description to the set of relevant and non-relevant queries. Vrajitoru worked with the vector space model, and thus document descriptions are real vectors in the interval which represent the weights associated to each term. Each document has associated just one description which leads to encode the whole collection in a single chromosome. The main problem of this model is that the fitness function only considers one query, and thus the document descriptions are adapted to match with this single query and not with a set of queries as in Gordon's model.

d. Document Clustering

While query expansion aims at expanding the number of relevant documents attracted by a given query, there are some other retrieval techniques that discover relationships among hopefully relevant documents independently of any query. An example of these techniques is document clustering. These techniques base their effectiveness on the Cluster Hypothesis: the documents which are relevant to the same query tend to be strongly related to each other. Robertson and Willet's proposed the idea to look for groups of terms appearing with similar frequencies in the documents of a collection In , Gordon designs a philosophy according to which it is possible to make a user-oriented clustering of documents using any classical clustering technique. The basic idea is that the system adapts document descriptions throughout time. In this way, documents being relevant to a query finally have similar descriptions and the system periodically clusters the new adapted descriptions. Gordon worked with the data base used in and considers the relative density of the cluster as goodness measure.

e. Query Formulation

Query formulation is an essential part of successful information retrieval. Query formulation deals with how well we are framing the user query. The three main factors affecting query formulation are media expertise, domain expertise and type of search. One retrieval technique prominently used to improve recall is called query expansion. Query expansion takes a query as input and adds new terms to it thereby producing a new, expanded query. The expansion terms may originate directly from thesauri, dictionaries, or end users. Abdelmgeid A.Aly proposed adaptive method using genetic algorithm to modify user's queries, based on relevance judgments. The algorithm shows the effects of applying GA to improve the effectiveness of queries in IR systems. The goal is to retrieve most relevant documents with less number of nonrelevant documents with respect to user's query in information retrieval system using genetic algorithm. The proposed GA approach gives better results than classical IR system when tested. Chen et al. used a GA as an Inductive Query By Example (IQBE) technique to learn the query terms ie for team learning that better represent a relevant document set provided by a user. They consider an information retrieval system based on the vector space model. Each chromosome is a fixed size, binary vector where each position is associated to an existing term in the initial relevant document set. Genetic operators are one-point crossover, uniform mutation and roulette wheel selection. Hao Lang, Bin Wang, Gareth Jones, Jin-Tao Li, Fan Ding and Yi-Xuan Liu proposed a new statistical method called Covering Topic Score (CTS) to predict the performance of the query for information retrieval. Estimation is based on how well the topic of user's query is covered by documents retrieved from a certain retrieval system. The goal of Aris Anagnostopoulos , Andrei Broder, Kunal Punera is to build the "best" short query that characterizes a document class using operators available within search engines.

Good classification accuracy can be achieved on average over multiple classes by queries with as few as ten terms. The proposed work shows that optimizing the efficiency of query execution by careful selection of terms can further reduce the query costs and also how classification can be performed effectively and efficiently using a search-engine model.

f. Match Function Adaptation

The aim is to use an GA to generate a similarity measure for a vector space information retrieval system to improve its retrieval efficacy for an specific user. This constitutes a new

relevance feedback since matching functions are adapted instead of queries. In Pathak et al. propose a new weighted matching function, which is the linear combination of different existing similarity functions. The weighting parameters are estimated by a GA based on relevance feedback from users. They used real coding, a classical generational scheme, twopoint crossover and Gaussian noise mutation. values. The rough value consists of an upper bound and a lower bound. Variables such as daily temperature are associated with a set of values instead of a single value. The upper and lower bounds of the set can represent variables using rough values. Rough equivalents of basic notions such as gene and chromosomes are defined. Two genetic operators in rough GA namely, *union* and *intersection* is also discussed which helps in the rough computing. These rough genetic operators provide additional flexibility for creating new generations during the evolution. Two new evaluation measures, called *precision* and *distance*, are also defined. The precision function quantifies information contained in a rough chromosome, while the distance function is used to calculate the dissimilarity between two rough chromosomes. A simple document retrieval example was used to demonstrate the usefulness of RGAs. Rough genetic algorithms seem to provide useful extensions for practical applications.

CHAPTER 8

CODES

1. ADMIN.PHP

```
<?php

class Gac{

        const POPULATION_SIZE = 20;
        const MUTATION_RATE = 10;
        const MUTATION_SIZE = 100;
        const SURVIVORS = 10;
        const SUCCESS_RATE = 1;
        const LETTERS = "abcdefghijklmnopqrstuvwxyz ";
        const OFFSPRING = 2;

        public function Gac($text)
        {
                $this->text = $text;
                $this->lenght = strlen($this->text);
                $this->population = 0;
                $this->max_rate = 0;
                $this->avarage_suc_rate = 0;
                $this->best = array();
                $this->passed = FALSE;
                $this->generations = 0;
        }

        private function check_success($data)
        {
                $rate = 0;

                for($i=0;$i<$this->lenght;$i++)
                {
                        $original_letter = substr($this->text,$i,1);
                        $text_letter = substr($data,$i,1);
                        if($original_letter == $text_letter)
                        {
                                $rate++;
                        }
                }

                $rate = $rate/$this->lenght;

                return $rate;
        }

        private function first_population($gene_code)
        {
```

```
                $words = array();
                for($i=0;$i<self::POPULATION_SIZE;$i++)
                {
                        $this->population++;
                        $mutation = TRUE; //this is the first
population, so mutate everything

                        $new_ind = $this-
>generate($gene_code,$mutation);
                        $words[$i]["word"] = $new_ind["word"];
                        $words[$i]["rate"] = $new_ind["rate"];
                        $words[$i]["gene_code"] =
$new_ind["gene_code"];

                        echo $new_ind["word"] . " - " .
$new_ind["rate"] ."<br>\n";

                }
                echo "end of first generation <br>\n";
                echo "----------<br>\n";
                array_unique($words);
                usort($words, "cmp");
                return $words;
        }

        private function kill($genes)
        {
                if(count($genes)>self::SURVIVORS)
                {
                        $diff = count($genes) - self::SURVIVORS;

                        for($i=0;$i<$diff;$i++)
                        {
                                array_pop($genes);
                        }

                        echo "killed $diff individuals<br>\n";
                }

                return $genes;
        }

        private function populate($genes)
        {
                $this->generations++;
                $new_genes = array();
                echo "------------ START OF " . $this->generations .
" th GENERATION ----------<br>\n";
                //cross everything
                usort($this->best,"cmp");
```

```
                for($j=0;$j<self::OFFSPRING;$j++)
                {
                        for($i=0;$i<count($genes);$i++)
                        {
                                $this->population++;
                                $best = FALSE;

                                if($genes[$i]["rate"]>=$this-
>max_rate)
                                {
                                        $new_gene                =
$genes[$i]["gene_code"];
                                        $best = TRUE;
                                        echo "stay still<br>";
                                }
                                else
                                {

        if($genes[$i]["gene_code"]!==$genes[$i+1]["gene_code"]
&& $genes[$i+1]["gene_code"])
                                        {
                                                $new_gene = $this-
>cross($genes[$i]["gene_code"],$genes[$i+1]["gene_code"]);
                                        }
                                        else
if($genes[$i]["gene_code"]!==$genes[$i+2]["gene_code"]      &&
$genes[$i+2]["gene_code"])
                                        {
                                                $new_gene = $this-
>cross($genes[$i]["gene_code"],$genes[$i+2]["gene_code"]);
                                        }
                                        else
                                        {
                                                $new_gene        =
$genes[$i]["gene_code"];
                                                echo          "crossed
nothing!<br>\n";
                                        }
                                }


        if($new_gene==$genes[$i]["gene_code"] && $i!==0)
                                {
                                        $new_gene      =      $this-
>mutate($new_gene);
                                        echo                   "<span
style='color:red'>mutated unexpectedly!</span><br>\n";
                                }

                                $mutation = FALSE;
```

```
                    if($this->population % self::MUTATION_SIZE == 0)
                    {
                        //mutate the new created gene if the mutation size reached
                        echo "mutation<br>\n";
                        $mutation = TRUE;
                    }

                    $new_ind = $this->generate($new_gene,$mutation);
                    $new_genes[$i]["word"] = $new_ind["word"];
                    $new_genes[$i]["rate"] = $new_ind["rate"];
                    $new_genes[$i]["gene_code"] = $new_ind["gene_code"];

                    if($new_ind["rate"]>=$this->max_rate && !in_array($new_ind,$this->best) )
                    {
                        $this->max_rate = $new_ind["rate"];
                        $this->best[] = $new_ind;
                        usort($this->best,"cmp");
                    }

                    echo $new_ind["word"] . " - " . $new_ind["rate"] ."<br>\n";
                }
            }


            array_unique($new_genes);
            usort($new_genes, "cmp");
            return $new_genes;
        }

        private function cross($gene1,$gene2)
        {
            $new_gene = array();
            for($i=0;$i<$this->lenght;$i++)
            {
                $rand = rand(0,1);
                if($rand)
                {
                    $new_gene[$i] = $gene1[$i];
                }
                else
```

```
                {
                    $new_gene[$i] = $gene2[$i];
                }
            }

            if($new_gene !== $gene1 && $new_gene !== $gene2)
            {
                echo "<span style='color:green'>cross successfull </span><br>";
            }

            return $new_gene;


        public function execute()
        {
            //the first gene code
            $gene_code = $this->generate_gene_code();
            $first_population = $this->first_population($gene_code);
            $result = $this->evolution($first_population);

            return $result;
        }

        private function evolution($genes_data)
        {
            //return $genes_data;

        if($genes_data[0]["rate"]>=self::SUCCESS_RATE)
            {
                echo $genes_data[0]["word"]. " found after ". $this->population . " try";
                return $genes_data;
            }
            else
            {
                if(count($this->best)>self::SURVIVORS || $this->passed)
                {
                    usort($this->best,"cmp");
                    $this->best = $this->kill($this->best);

                    $genes = $this->best;
                    $this->passed = TRUE;
                }
                else
                {
                    $genes = $genes_data;
```

```
                }
                return $this->evolution($this->populate($genes));
            }
        }

        private function generate_gene_code()
        {
            $code = array();
            for($i=0;$i<$this->lenght;$i++)
            {
                $code[] = rand(0,strlen(self::LETTERS));
            }
            return $code;
        }

        private function generate($gene_code,$mutation)
        {
            //generates new individuals according to genetic code
            if($mutation)
            {
                $gene_code = $this->mutate($gene_code);
            }

            $new_word = "";

            for($i=0;$i<$this->lenght;$i++)
            {
                $letter = substr(self::LETTERS,$gene_code[$i],1);
                $new_word .= $letter;
            }

            $new_ind = array();

            $new_ind["gene_code"] = $gene_code;
            $new_ind["word"] = $new_word;
            $new_ind["rate"] = $this->check_success($new_word);

            return $new_ind;
        }

        private function mutate($gene_code)
        {
            //mutate original data and returns it
            $mutate_letter_num = floor(self::MUTATION_RATE * $this->lenght / 100);
            $mutate_letters = array();
```

```
                for($i=0;$i<$mutate_letter_num;$i++)
                {
                        $which = rand(0,$this->lenght-1);
                        $mutate_letters[] = $which;
                }

                array_unique($mutate_letters);

                foreach($mutate_letters as $letter_num)
                {
                        $gene_code[$letter_num]                    =
rand(0,strlen(self::LETTERS));
                }
                return $gene_code;
        }
}

?>
```

2. DATABASE.PHP

```
<?php

class Gac{

        const POPULATION_SIZE = 20;
        const MUTATION_RATE = 10;
        const MUTATION_SIZE = 100;
        const SURVIVORS = 10;
        const SUCCESS_RATE = 1;
        const LETTERS = "abcdefghijklmnopqrstuvwxyz ";
        const OFFSPRING = 2;

        public function Gac($text)
        {
                $this->text = $text;
                $this->lenght = strlen($this->text);
                $this->population = 0;
                $this->max_rate = 0;
                $this->avarage_suc_rate = 0;
                $this->best = array();
                $this->passed = FALSE;
                $this->generations = 0;
        }

        private function check_success($data)
        {
                $rate = 0;

                for($i=0;$i<$this->lenght;$i++)
```

```
                {
                        $original_letter = substr($this->text,$i,1);
                        $text_letter = substr($data,$i,1);
                        if($original_letter == $text_letter)
                        {
                                $rate++;
                        }
                }

                $rate = $rate/$this->lenght;

                return $rate;
        }

        private function first_population($gene_code)
        {
                $words = array();
                for($i=0;$i<self::POPULATION_SIZE;$i++)
                {
                        $this->population++;
                        $mutation = TRUE; //this is the first
population, so mutate everything

                        $new_ind = $this-
>generate($gene_code,$mutation);
                        $words[$i]["word"] = $new_ind["word"];
                        $words[$i]["rate"] = $new_ind["rate"];
                        $words[$i]["gene_code"] =
$new_ind["gene_code"];

                        echo $new_ind["word"] . " - " .
$new_ind["rate"] ."<br>\n";

                }
                echo "end of first generation <br>\n";
                echo "----------<br>\n";
                array_unique($words);
                usort($words, "cmp");
                return $words;
        }

        private function kill($genes)
        {
                if(count($genes)>self::SURVIVORS)
                {
                        $diff = count($genes) - self::SURVIVORS;

                        for($i=0;$i<$diff;$i++)
                        {
                                array_pop($genes);
```

```
                }

                echo "killed $diff individuals<br>\n";
            }

            return $genes;
        }

        private function populate($genes)
        {
            $this->generations++;
            $new_genes = array();
            echo "---------- START OF " . $this->generations .
" th GENERATION ----------<br>\n";
            //cross everything
            usort($this->best,"cmp");
            for($j=0;$j<self::OFFSPRING;$j++)
            {
                for($i=0;$i<count($genes);$i++)
                {
                    $this->population++;
                    $best = FALSE;

                    if($genes[$i]["rate"]>=$this-
>max_rate)
                    {
                        $new_gene                    =
$genes[$i]["gene_code"];
                        $best = TRUE;
                        echo "stay still<br>";
                    }
                    else
                    {

        if($genes[$i]["gene_code"]!==$genes[$i+1]["gene_code"]
&& $genes[$i+1]["gene_code"])
                        {
                            $new_gene = $this-
>cross($genes[$i]["gene_code"],$genes[$i+1]["gene_code"]);
                        }
                        else
if($genes[$i]["gene_code"]!==$genes[$i+2]["gene_code"]      &&
$genes[$i+2]["gene_code"])
                        {
                            $new_gene = $this-
>cross($genes[$i]["gene_code"],$genes[$i+2]["gene_code"]);
                        }
                        else
                        {
```

```
                    $new_gene =
$genes[$i]["gene_code"];
                    echo "crossed
nothing!<br>\n";
                }
            }

        if($new_gene==$genes[$i]["gene_code"] && $i!==0)
            {
                $new_gene = $this-
>mutate($new_gene);
                echo "<span
style='color:red'>mutated unexpectedly!</span><br>\n";
            }

            $mutation = FALSE;

            if($this->population %
self::MUTATION_SIZE == 0)
            {
                //mutate the new created
gene if the mutation size reached
                echo "mutation<br>\n";
                $mutation = TRUE;
            }

            $new_ind = $this-
>generate($new_gene,$mutation);
            $new_genes[$i]["word"] =
$new_ind["word"];
            $new_genes[$i]["rate"] =
$new_ind["rate"];
            $new_genes[$i]["gene_code"] =
$new_ind["gene_code"];

            if($new_ind["rate"]>=$this-
>max_rate && !in_array($new_ind,$this->best) )
            {
                $this->max_rate =
$new_ind["rate"];
                $this->best[] = $new_ind;
                usort($this->best,"cmp");
            }

            echo $new_ind["word"] . " - " .
$new_ind["rate"] ."<br>\n";
        }
    }
```

```
                array_unique($new_genes);
                usort($new_genes, "cmp");
                return $new_genes;
        }

        private function cross($gene1,$gene2)
        {
                $new_gene = array();
                for($i=0;$i<$this->lenght;$i++)
                {
                        $rand = rand(0,1);
                        if($rand)
                        {
                                $new_gene[$i] = $gene1[$i];
                        }
                        else
                        {
                                $new_gene[$i] = $gene2[$i];
                        }
                }
                if($new_gene !== $gene1 && $new_gene !== $gene2)
                {
                        echo "<span style='color:green'>cross successfull </span><br>";
                }
                return $new_gene;


        public function execute()
        {
                //the first gene code
                $gene_code = $this->generate_gene_code();
                $first_population = $this->first_population($gene_code);
                $result = $this->evolution($first_population);

                return $result;
        }

        private function evolution($genes_data)
        {
                //return $genes_data;

                if($genes_data[0]["rate"]>=self::SUCCESS_RATE)
                {
```

```
                        echo $genes_data[0]["word"]. " found after
". $this->population . " try";
                        return $genes_data;
                }
                else
                {
                        if(count($this->best)>self::SURVIVORS ||
$this->passed)
                        {
                                usort($this->best,"cmp");
                                $this->best = $this->kill($this-
>best);
                                $genes = $this->best;
                                $this->passed = TRUE;
                        }
                        else
                        {
                                $genes = $genes_data;
                        }
                        return $this->evolution($this-
>populate($genes));
                }
        }

        private function generate_gene_code()
        {
                $code = array();
                for($i=0;$i<$this->lenght;$i++)
                {
                        $code[] = rand(0,strlen(self::LETTERS));
                }
                return $code;
        }

        private function generate($gene_code,$mutation)
        {
                //generates new individuals according to genetic
code
                if($mutation)
                {
                        $gene_code = $this->mutate($gene_code);
                }

                $new_word = "";

                for($i=0;$i<$this->lenght;$i++)
                {
                        $letter =
substr(self::LETTERS,$gene_code[$i],1);
                        $new_word .= $letter;
```

```
                }

                $new_ind = array();

                $new_ind["gene_code"] = $gene_code;
                $new_ind["word"] = $new_word;
                $new_ind["rate"]                =                $this-
>check_success($new_word);

                return $new_ind;
        }

        private function mutate($gene_code)
        {
                //mutate original data and returns it
                $mutate_letter_num                                              =
floor(self::MUTATION_RATE * $this->lenght / 100);
                $mutate_letters = array();

                for($i=0;$i<$mutate_letter_num;$i++)
                {
                        $which = rand(0,$this->lenght-1);
                        $mutate_letters[] = $which;
                }

                array_unique($mutate_letters);

                foreach($mutate_letters as $letter_num)
                {                                                               =
                        $gene_code[$letter_num]
rand(0,strlen(self::LETTERS));
                }
                return $gene_code;
        }
}

?>
```

3. SPIDER.PHP

```
<?php
    set_time_limit (0);
        $include_dir = "../include";
        include "auth.php";
        require_once ("$include_dir/commonfuncs.php");
        $all = 0;
        extract (getHttpVars());
        $settings_dir =  "../settings";
        require_once ("$settings_dir/conf.php");
        include "messages.php";
        include "spiderfuncs.php";
```

```
error_reporting (E_ALL ^ E_NOTICE ^ E_WARNING);


$delay_time = 0;

if (isset($soption) && $soption == 'full') {
        $maxlevel = -1;

}

if (!isset($domaincb)) {
        $domaincb = 0;

}

if(!isset($reindex)) {
        $reindex=0;
}

if(!isset($maxlevel)) {
        $maxlevel=0;
}


if ($keep_log) {
        if ($log_format=="html") {
                $log_file =
$log_dir."/".Date("ymdHi").".html";
        } else {
                $log_file =
$log_dir."/".Date("ymdHi").".log";
        }

        if (!$log_handle = fopen($log_file, 'w')) {
                die ("Logging option is set, but cannot open
file for logging.");
        }
}

if ($all == 1) {
        index_all();
} else {

        if ($reindex == 1 && $command_line == 1) {
                $result=mysql_query("select url,
spider_depth, required, disallowed, can_leave_domain from
".$mysql_table_prefix."sites where url='$url'");
                echo mysql_error();
                if($row=mysql_fetch_row($result)) {
                        $url = $row[0];
```

```
                $maxlevel = $row[1];
                $in= $row[2];
                $out = $row[3];
                $domaincb = $row[4];
                if ($domaincb=='') {
                        $domaincb=0;
                }
                if ($maxlevel == -1) {
                        $soption = 'full';
                } else {
                        $soption = 'level';
                }
        }

    }
    if (!isset($in)) {
            $in = "";
    }
    if (!isset($out)) {
            $out = "";
    }

    index_site($url, $reindex, $maxlevel, $soption, $in,
$out, $domaincb);

}

$tmp_urls  = Array();


function microtime_float(){
  list($usec, $sec) = explode(" ", microtime());
  return ((float)$usec + (float)$sec);
}


function index_url($url, $level, $site_id, $md5sum,
$domain, $indexdate, $sessid, $can_leave_domain, $reindex) {
        global $entities, $min_delay;
        global $command_line;
        global $min_words_per_page;
        global $supdomain;
        global $mysql_table_prefix, $user_agent,
$tmp_urls, $delay_time, $domain_arr;
        $needsReindex = 1;
        $deletable = 0;

        $url_status = url_status($url);
        $thislevel = $level - 1;
```

```
                if (strstr($url_status['state'], "Relocation")) {
                        $url = preg_replace("/ /", "",
url_purify($url_status['path'], $url, $can_leave_domain));

                        if ($url <> '') {
                                $result = mysql_query("select link
from ".$mysql_table_prefix."temp where link='$url' && id =
'$sessid'");
                                echo mysql_error();
                                $rows = mysql_numrows($result);
                                if ($rows == 0) {
                                        mysql_query ("insert into
".$mysql_table_prefix."temp (link, level, id) values ('$url', '$level',
'$sessid')");
                                        echo mysql_error();
                                }
                        }

                        $url_status['state'] == "redirected";
                }
                ini_set("user_agent", $user_agent);
                if ($url_status['state'] == 'ok') {
                        $OKtoIndex = 1;
                        $file_read_error = 0;

                        if (time() - $delay_time < $min_delay) {
                                sleep ($min_delay- (time() -
$delay_time));
                        }
                        $delay_time = time();
                        if (!fst_lt_snd(phpversion(), "4.3.0")) {
                                $file = file_get_contents($url);
                                if ($file === FALSE) {
                                        $file_read_error = 1;
                                }
                        } else {
                                $fl = @fopen($url, "r");
                                if ($fl) {
                                        while ($buffer = @fgets($fl,
4096)) {
                                                $file .= $buffer;
                                        }
                                } else {
                                        $file_read_error = 1;
                                }

                                fclose ($fl);
                        }
                        if ($file_read_error) {
                                $contents = getFileContents($url);
```

```
                $file = $contents['file'];
            }

            $pageSize =
number_format(strlen($file)/1024, 2, ".", "");
            printPageSizeReport($pageSize);

            if ($url_status['content'] != 'text') {
                $file = extract_text($file,
$url_status['content']);
            }

            printStandardReport('starting',
$command_line);

            $newmd5sum = md5($file);

            if ($md5sum == $newmd5sum) {

        printStandardReport('md5notChanged',$command_line);
                $OKtoIndex = 0;
            } else if (isDuplicateMD5($newmd5sum)) {
                $OKtoIndex = 0;

        printStandardReport('duplicate',$command_line);
            }

            if (($md5sum != $newmd5sum || $reindex
==1) && $OKtoIndex == 1) {
                $urlparts = parse_url($url);
                $newdomain = $urlparts['host'];
                $type = 0;

                $data = clean_file($file, $url,
$url_status['content']);

                if ($data['noindex'] == 1) {
                    $OKtoIndex = 0;
                    $deletable = 1;

        printStandardReport('metaNoindex',$command_line);
                }

                $wordarray =
unique_array(explode(" ", $data['content']));
```

```
                    if ($data['nofollow'] != 1) {
                        $links = get_links($file, $url,
$can_leave_domain, $data['base']);
                        $links =
distinct_array($links);

                        $all_links = count($links);
                        $numoflinks = 0;

                        if (is_array($links)) {
                            reset ($links);

                            while ($thislink =
each($links)) {

                                if
($tmp_urls[$thislink[1]] != 1) {

                                    $tmp_urls[$thislink[1]] = 1;

                                    $numoflinks++;

                                    mysql_query ("insert into ".$mysql_table_prefix."temp
(link, level, id) values ('$thislink[1]', '$level', '$sessid')");
                                    echo
mysql_error();

                                }
                            }
                        }
                    } else {

                        printStandardReport('noFollow',$command_line);
                    }

                    if ($OKtoIndex == 1) {

                        $title = $data['title'];
                        $host = $data['host'];
                        $path = $data['path'];
                        $fulltxt = $data['fulltext'];
                        $desc =
substr($data['description'], 0,254);
                        $url_parts = parse_url($url);
                        $domain_for_db =
$url_parts['host'];

                        if
(isset($domain_arr[$domain_for_db])) {
                            $dom_id =
$domain_arr[$domain_for_db];
                        } else {
```

```
                                        mysql_query("insert
into ".$mysql_table_prefix."domains (domain) values
('$domain_for_db')");
                                        $dom_id =
mysql_insert_id();

        $domain_arr[$domain_for_db] = $dom_id;
                                    }

                                    $wordarray = calc_weights
($wordarray, $title, $host, $path, $data['keywords']);

                                    //if there are words to index,
add the link to the database, get its id, and add the word + their
relation
                                    if (is_array($wordarray) &&
count($wordarray) > $min_words_per_page) {
                                        if ($md5sum == '') {
                                            mysql_query
("insert into ".$mysql_table_prefix."links (site_id, url, title,
description, fulltxt, indexdate, size, md5sum, level) values
('$site_id', '$url', '$title', '$desc', '$fulltxt', curdate(), '$pageSize',
'$newmd5sum', $thislevel)");
                                            echo
mysql_error();
                                            $result =
mysql_query("select link_id from ".$mysql_table_prefix."links
where url='$url'");
                                            echo
mysql_error();
                                            $row =
mysql_fetch_row($result);
                                            $link_id =
$row[0];

        save_keywords($wordarray, $link_id, $dom_id);

        printStandardReport('indexed', $command_line);
                                        }else if (($md5sum
<> '') && ($md5sum <> $newmd5sum)) { //if page has changed,
start updating

                                            $result =
mysql_query("select link_id from ".$mysql_table_prefix."links
where url='$url'");
                                            echo
mysql_error();
```


Learning Resource Center
Acc. No. SP09008
JUIT Waknaghat, Solan (H.P.)

```
                                        $row =
mysql_fetch_row($result);
                                        $link_id =
$row[0];
                                        for
($i=0;$i<=15; $i++) {
                                            $char
= dechex($i);
        mysql_query ("delete from
".$mysql_table_prefix."link_keyword$char where
link_id=$link_id");
                                        echo
mysql_error();
                                    }
        save_keywords($wordarray, $link_id, $dom_id);
                                        $query =
"update ".$mysql_table_prefix."links set title='$title', description
='$desc', fulltxt = '$fulltxt', indexdate=now(), size = '$pageSize',
md5sum='$newmd5sum', level=$thislevel where
link_id=$link_id";
        mysql_query($query);
                                        echo
mysql_error();
        printStandardReport('re-indexed', $command_line);
                                    }
                                }else {
        printStandardReport('minWords', $command_line);
                                }
                            }
                        }
                    } else {
                        $deletable = 1;
                        printUrlStatus($url_status['state'],
$command_line);
                    }
                    if ($reindex ==1 && $deletable == 1) {
                        check_for_removal($url);
                    } else if ($reindex == 1) {
                    }
                    if (!isset($all_links)) {
                        $all_links = 0;
                    }
```

```
                if (!isset($numoflinks)) {
                        $numoflinks = 0;
                }
                printLinksReport($numoflinks, $all_links,
$command_line);
        }


        function index_site($url, $reindex, $maxlevel, $soption,
$url_inc, $url_not_inc, $can_leave_domain) {
                global $mysql_table_prefix, $command_line,
$mainurl, $tmp_urls, $domain_arr, $all_keywords;
                if (!isset($all_keywords)) {
                        $result = mysql_query("select keyword_ID,
keyword from ".$mysql_table_prefix."keywords");
                        echo mysql_error();
                        while($row=mysql_fetch_array($result)) {
                                $all_keywords[addslashes($row[1])]
= $row[0];
                        }
                }
                $compurl = parse_url($url);
                if ($compurl['path'] == '')
                        $url = $url . "/";

                $t = microtime();
                $a = getenv("REMOTE_ADDR");
                $sessid = md5 ($t.$a);


                $urlparts = parse_url($url);

                $domain = $urlparts['host'];
                if (isset($urlparts['port'])) {
                        $port = (int)$urlparts['port'];
                }else {
                        $port = 80;
                }



                $result = mysql_query("select site_id from
".$mysql_table_prefix."sites where url='$url'");
                echo mysql_error();
                $row = mysql_fetch_row($result);
                $site_id = $row[0];

                if ($site_id != "" && $reindex == 1) {
```

```
                    mysql_query ("insert into
".$mysql_table_prefix."temp (link, level, id) values ('$url', 0,
'$sessid')");
                    echo mysql_error();
                    $result = mysql_query("select url, level
from ".$mysql_table_prefix."links where site_id = $site_id");
                    while ($row = mysql_fetch_array($result)) {
                        $site_link = $row['url'];
                        $link_level = $row['level'];
                        if ($site_link != $url) {
                            mysql_query ("insert into
".$mysql_table_prefix."temp (link, level, id) values ('$site_link',
$link_level, '$sessid')");
                        }
                    }

                    $qry = "update ".$mysql_table_prefix."sites
set indexdate=now(), spider_depth = $maxlevel, required =
'$url_inc'," .
                        "disallowed = '$url_not_inc',
can_leave_domain=$can_leave_domain where site_id=$site_id";
                    mysql_query ($qry);
                    echo mysql_error();
                } else if ($site_id == '') {
                    mysql_query ("insert into
".$mysql_table_prefix."sites (url, indexdate, spider_depth,
required, disallowed, can_leave_domain) " .
                        "values ('$url', now(),
$maxlevel, '$url_inc', '$url_not_inc', $can_leave_domain)");
                    echo mysql_error();
                    $result = mysql_query("select site_ID from
".$mysql_table_prefix."sites where url='$url'");
                    $row = mysql_fetch_row($result);
                    $site_id = $row[0];
                } else {
                    mysql_query ("update
".$mysql_table_prefix."sites set indexdate=now(), spider_depth =
$maxlevel, required = '$url_inc'," .
                        "disallowed = '$url_not_inc',
can_leave_domain=$can_leave_domain where site_id=$site_id");
                    echo mysql_error();
                }


                $result = mysql_query("select site_id, temp_id,
level, count, num from ".$mysql_table_prefix."pending where
site_id='$site_id'");
                echo mysql_error();
                $row = mysql_fetch_row($result);
                $pending = $row[0];
```

```
		$level = 0;
		$domain_arr = get_domains();
		if ($pending == '') {
			mysql_query ("insert into
".$mysql_table_prefix."temp (link, level, id) values ('$url', 0,
'$sessid')");
			echo mysql_error();
		} else if ($pending != '') {

	printStandardReport('continueSuspended',$command_line)
;
			mysql_query("select temp_id, level, count
from ".$mysql_table_prefix."pending where site_id='$site_id'");
			echo mysql_error();
			$sessid = $row[1];
			$level = $row[2];
			$pend_count = $row[3] + 1;
			$num = $row[4];
			$pending = 1;
			$tmp_urls = get_temp_urls($sessid);
		}

		if ($reindex != 1) {
			mysql_query ("insert into
".$mysql_table_prefix."pending (site_id, temp_id, level, count)
values ('$site_id', '$sessid', '0', '0')");
			echo mysql_error();
		}


		$time = time();


		$omit = check_robot_txt($url);

		printHeader ($omit, $url, $command_line);


		$mainurl = $url;
		$num = 0;

		while (($level <= $maxlevel && $soption ==
'level') || ($soption == 'full')) {
			if ($pending == 1) {
				$count = $pend_count;
				$pending = 0;
			} else
				$count = 0;

			$links = array();
```

```
                $result = mysql_query("select distinct link
from ".$mysql_table_prefix."temp where level=$level &&
id='$sessid' order by link");
                echo mysql_error();
                $rows = mysql_num_rows($result);

                if ($rows == 0) {
                        break;
                }

                $i = 0;

                while ($row = mysql_fetch_array($result)) {
                        $links[] = $row['link'];
                }

                reset ($links);


                while ($count < count($links)) {
                        $num++;
                        $thislink = $links[$count];
                        $urlparts = parse_url($thislink);
                        reset ($omit);
                        $forbidden = 0;
                        foreach ($omit as $omiturl) {
                                $omiturl = trim($omiturl);

                                $omiturl_parts                =
parse_url($omiturl);

                                if    ($omiturl_parts['scheme']
== '') {

                                        $check_omit           =
$urlparts['host'] . $omiturl;

                                } else {
                                        $check_omit           =
$omiturl;

                                }

                                if              (strpos($thislink,
$check_omit)) {

        printRobotsReport($num, $thislink, $command_line);

        check_for_removal($thislink);
                                        $forbidden = 1;
                                        break;
                                }
                        }
```

```
                    if (!check_include($thislink,
$url_inc, $url_not_inc )) {
                        printUrlStringReport($num,
$thislink, $command_line);
        check_for_removal($thislink);
                        $forbidden = 1;
                    }

                    if ($forbidden == 0) {
                        printRetrieving($num,
$thislink, $command_line);
                        $query = "select md5sum,
indexdate from ".$mysql_table_prefix."links where
url='$thislink'";
                        $result =
mysql_query($query);
                        echo mysql_error();
                        $rows =
mysql_num_rows($result);
                        if ($rows == 0) {
                            index_url($thislink,
$level+1, $site_id, '', $domain, '', $sessid, $can_leave_domain,
$reindex);

                            mysql_query("update
".$mysql_table_prefix."pending set level = $level, count=$count,
num=$num where site_id=$site_id");
                            echo mysql_error();
                        }else if ($rows <> 0 &&
$reindex == 1) {
                            $row =
mysql_fetch_array($result);
                            $md5sum =
$row['md5sum'];
                            $indexdate =
$row['indexdate'];
                            index_url($thislink,
$level+1, $site_id, $md5sum, $domain, $indexdate, $sessid,
$can_leave_domain, $reindex);
                            mysql_query("update
".$mysql_table_prefix."pending set level = $level, count=$count,
num=$num where site_id=$site_id");
                            echo mysql_error();
                        }else {
        printStandardReport('inDatabase',$command_line);
                        }
```

```
                        }
                        $count++;
                    }
                    $level++;
                }

                mysql_query ("delete from
".$mysql_table_prefix."temp where id = '$sessid'");
                echo mysql_error();
                mysql_query ("delete from
".$mysql_table_prefix."pending where site_id = '$site_id'");
                echo mysql_error();
                printStandardReport('completed',$command_line);


        }

        function index_all() {
                global $mysql_table_prefix;
                $result=mysql_query("select url, spider_depth,
required, disallowed, can_leave_domain from
".$mysql_table_prefix."sites");
                echo mysql_error();
        while ($row=mysql_fetch_row($result)) {
                $url = $row[0];
                        $depth = $row[1];
                $include = $row[2];
                $not_include = $row[3];
                $can_leave_domain = $row[4];
                if ($can_leave_domain=='') {
                        $can_leave_domain=0;
                }
                if ($depth == -1) {
                        $soption = 'full';
                } else {
                        $soption = 'level';
                }
                        index_site($url, 1, $depth, $soption,
$include, $not_include, $can_leave_domain);
                }
        }

        function get_temp_urls ($sessid) {
                global $mysql_table_prefix;
                $result = mysql_query("select link from
".$mysql_table_prefix."temp where id='$sessid'");
                echo mysql_error();
                $tmp_urls = Array();
        while ($row=mysql_fetch_row($result)) {
                        $tmp_urls[$row[0]] = 1;
```

```
                }
                return $tmp_urls;

        }

        function get_domains () {
                global $mysql_table_prefix;
                $result = mysql_query("select domain_id, domain
 from ".$mysql_table_prefix."domains");
                echo mysql_error();
                $domains = Array();
        while ($row=mysql_fetch_row($result)) {
                        $domains[$row[1]] = $row[0];
                }
                return $domains;

        }

        function commandline_help() {
                print "Usage: php spider.php <options>\n\n";
                print "Options:\n";
                print " -all\t\t Reindex everything in the
database\n";
                print " -u <url>\t Set url to index\n";
                print " -f\t\t Set indexing depth to full (unlimited
depth)\n";
                print " -d <num>\t Set indexing depth to <num>\n";
                print " -l\t\t Allow spider to leave the initial
domain\n";
                print " -r\t\t Set spider to reindex a site\n";
                print " -m <string>\t Set the string(s) that an url
must include (use \\n as a delimiter between multiple strings)\n";
                print " -n <string>\t Set the string(s) that an url must
not include (use \\n as a delimiter between multiple strings)\n";
        }

        printStandardReport('quit',$command_line);
        if ($email_log) {
                $indexed = ($all==1) ? 'ALL' : $url;
                $log_report = "";
                if ($log_handle) {
                        $log_report = "Log saved into $log_file";
                }
                mail($admin_email, "Sphider indexing report",
"Sphider has finished indexing $indexed at ".date("y-m-d H:i:s").".
".$log_report);
        }
        if ( $log_handle) {
                fclose($log_handle);
        }
```

```
?>
```

4. AUTHENTICATION.PHP

```
<?php
        error_reporting(E_ERROR | E_PARSE);
        $admin = "admin";
        $admin_pw = "admin";

        session_start();

if (isset($_POST['user']) && isset($_POST['pass'])) {

        $username = $_POST['user'];
        $password = $_POST['pass'];
        if ($username == $admin) && ($password
==$admin_pw)) {
                $_SESSION['admin'] = $username;
                $_SESSION['admin_pw'] = $password;
        }
        header("Location: admin.php");
}       elseif ((isset($_SESSION['admin']) &&
isset($_SESSION['admin_pw']) &&$_SESSION['admin'] ==
$admin && $_SESSION['admin_pw'] == $admin_pw ) ||
(getenv("REMOTE_ADDR")=="")) {

} else {

        ?>
        <html>
        <head>
        <title>GASE Admin Login</title>
                <LINK REL=STYLESHEET HREF="admin.css"
TYPE="text/css">
        </head>

        <body>
        <center>
        <br><br>

        <fieldset style="width:30%;"><legend><b>GASE Admin
Login</b></legend>
        <form action="auth.php" method="post">

        <table>
        <tr><td>Username</td><td><input type="text"
name="user"></td></tr>
        <tr><td>Password</td><td><input type="password"
name="pass"></td></tr>
```

```
        <tr><td></td><td><input type="submit" value="Log in"
id="submit"></td>
        </tr></table>
        </form>
        </fieldset>
        </center>
        </body>
        </html>
        <?php
        exit();
}


$settings_dir = "../settings";
include "$settings_dir/database.php";

?>
```

5. GENETIC ALGO.PHP

```
<?php

class Gac{

        const POPULATION_SIZE = 20;
        const MUTATION_RATE = 10;
        const MUTATION_SIZE = 100;
        const SURVIVORS = 10;
        const SUCCESS_RATE = 1;
        const LETTERS = "abcdefghijklmnopqrstuvwxyz ";
        const OFFSPRING = 2;

        public function Gac($text)
        {
                $this->text = $text;
                $this->lenght = strlen($this->text);
                $this->population = 0;
                $this->max_rate = 0;
                $this->avarage_suc_rate = 0;
                $this->best = array();
                $this->passed = FALSE;
                $this->generations = 0;
        }

        private function check_success($data)
        {
                $rate = 0;

                for($i=0;$i<$this->lenght;$i++)
                {
                        $original_letter = substr($this->text,$i,1);
```

```
                $text_letter = substr($data,$i,1);
                if($original_letter == $text_letter)
                {
                        $rate++;
                }
        }

        $rate = $rate/$this->lenght;

        return $rate;
}

private function first_population($gene_code)
{
        $words = array();
        for($i=0;$i<self::POPULATION_SIZE;$i++)
        {
                $this->population++;
                $mutation = TRUE; //this is the first
population, so mutate everything

                $new_ind = $this-
>generate($gene_code,$mutation);
                $words[$i]["word"] = $new_ind["word"];
                $words[$i]["rate"] = $new_ind["rate"];
                $words[$i]["gene_code"] =
$new_ind["gene_code"];

                echo $new_ind["word"] . " - " .
$new_ind["rate"] ."<br>\n";

        }
        echo "end of first generation <br>\n";
        echo "----------<br>\n";
        array_unique($words);
        usort($words, "cmp");
        return $words;
}

private function kill($genes)
{
        if(count($genes)>self::SURVIVORS)
        {
                $diff = count($genes) - self::SURVIVORS;

                for($i=0;$i<$diff;$i++)
                {
                        array_pop($genes);
                }
```

```
                    echo "killed $diff individuals<br>\n";
                }

                return $genes;
        }

        private function populate($genes)
        {
                $this->generations++;
                $new_genes = array();
                echo "------------ START OF " . $this->generations .
" th GENERATION ----------------<br>\n";
                //cross everything
                usort($this->best,"cmp");
                for($j=0;$j<self::OFFSPRING;$j++)
                {
                        for($i=0;$i<count($genes);$i++)
                        {
                                $this->population++;
                                $best = FALSE;

                                if($genes[$i]["rate"]>=$this-
>max_rate)
                                {
                                        $new_gene                =
$genes[$i]["gene_code"];
                                        $best = TRUE;
                                        echo "stay still<br>";
                                }
                                else
                                {

        if($genes[$i]["gene_code"]!==$genes[$i+1]["gene_code"]
&& $genes[$i+1]["gene_code"])
                                        {
                                                $new_gene = $this-
>cross($genes[$i]["gene_code"],$genes[$i+1]["gene_code"]);
                                        }
                                        else
if($genes[$i]["gene_code"]!==$genes[$i+2]["gene_code"]      &&
$genes[$i+2]["gene_code"])
                                        {
                                                $new_gene = $this-
>cross($genes[$i]["gene_code"],$genes[$i+2]["gene_code"]);
                                        }
                                        else
                                        {
                                                $new_gene        =
$genes[$i]["gene_code"];
```

```
                                        echo "crossed
nothing!<br>\n";
                                    }
                                }

                    if($new_gene==$genes[$i]["gene_code"] && $i!==0)
                                {
                                    $new_gene = $this-
>mutate($new_gene);
                                    echo "<span
style='color:red'>mutated unexpectedly!</span><br>\n";
                                }

                                $mutation = FALSE;

                                if($this->population %
self::MUTATION_SIZE == 0)
                                {
                                    //mutate the new created
gene if the mutation size reached
                                    echo "mutation<br>\n";
                                    $mutation = TRUE;
                                }

                                $new_ind = $this-
>generate($new_gene,$mutation);
                                $new_genes[$i]["word"] =
$new_ind["word"];
                                $new_genes[$i]["rate"] =
$new_ind["rate"];
                                $new_genes[$i]["gene_code"] =
$new_ind["gene_code"];

                                if($new_ind["rate"]>=$this-
>max_rate && !in_array($new_ind,$this->best) )
                                {
                                    $this->max_rate =
$new_ind["rate"];
                                    $this->best[] = $new_ind;
                                    usort($this->best,"cmp");
                                }

                                echo $new_ind["word"] . " - " .
$new_ind["rate"] ."<br>\n";
                            }
                        }


                    array_unique($new_genes);
```

```
                usort($new_genes, "cmp");
                return $new_genes;
        }

        private function cross($gene1,$gene2)
        {
                $new_gene = array();
                for($i=0;$i<$this->lenght;$i++)
                {
                        $rand = rand(0,1);
                        if($rand)
                        {
                                $new_gene[$i] = $gene1[$i];
                        }
                        else
                        {
                                $new_gene[$i] = $gene2[$i];
                        }
                }

                if($new_gene !== $gene1 && $new_gene !== $gene2)
                {
                        echo "<span style='color:green'>cross successfull </span><br>";
                }

                return $new_gene;


        public function execute()
        {
                //the first gene code
                $gene_code = $this->generate_gene_code();
                $first_population = $this->first_population($gene_code);
                $result = $this->evolution($first_population);

                return $result;
        }

        private function evolution($genes_data)
        {
                //return $genes_data;

        if($genes_data[0]["rate"]>=self::SUCCESS_RATE)
                {
                        echo $genes_data[0]["word"]. " found after ". $this->population . " try";
                        return $genes_data;
```

```
                }
                else
                {
                        if(count($this->best)>self::SURVIVORS ||
$this->passed)
                        {
                                usort($this->best,"cmp");
                                $this->best = $this->kill($this-
>best);
                                $genes = $this->best;
                                $this->passed = TRUE;
                        }
                        else
                        {
                                $genes = $genes_data;
                        }
                        return $this->evolution($this-
>populate($genes));
                }
        }

        private function generate_gene_code()
        {
                $code = array();
                for($i=0;$i<$this->lenght;$i++)
                {
                        $code[] = rand(0,strlen(self::LETTERS));
                }
                return $code;
        }

        private function generate($gene_code,$mutation)
        {
                //generates new individuals according to genetic
code
                if($mutation)
                {
                        $gene_code = $this->mutate($gene_code);
                }

                $new_word = "";

                for($i=0;$i<$this->lenght;$i++)
                {
                        $letter =
substr(self::LETTERS,$gene_code[$i],1);
                        $new_word .= $letter;
                }

                $new_ind = array();
```

(V)

REFERENCES

[1] International Journal of Computer and Electrical Engineering, Vol. 2, No. 1, February, 2010 1793-8163

First A. S.Siva Sathya , Second B. Philomina Simon, *Member IACSIT*

[2] Goldberg, D. E., Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, 1989.

[3] Vicente P., Cristina P., "Order-Based Fitness Functions for Ger.etic Algorithms Applied to Relevance Feedback", Journal Of The American Society For Information Science And Technology, 54(2):152–160, 2003.

[4] Ahmed A. A. Radwan, Bahgat A. Abdel Latef, Abdel Mgeid A. Ali, Osman A. Sadek, "Using Genetic Algorithm to Improve Information Retrieval Systems", proceedings of world academy of since, engineering and technology, volume 17, ISSN 1307-6884, 2006.

[5] M.Boughanem, C. Chrisment, L. Tamine, —Multiple query evaluation based on an enhanced genetic algorithm‖, Information Processing and Management 39,215–231, 2003.

Abdelmgeid A.Aly, Applying genetic algorithm in query

improvement problem, International journal "Information

Technologies and Knowledge" Vol 1, 309 - 316, 2007

[6] David E Goldberg ,Genetic Algorithms in Search, Optimization, Machine Learning , Addison Wesley , 1989

[7] Helen J. Peat and Peter Willett*,‖The Limitations of Term Cooccurrence Data for Query Expansion in Document Retrieval Systems‖, Journal of The American Society for Information Science. 42(5),378-383, 1991

[8] Lothar M. Schmitt ," Fundamental Study ,Theory of genetic algorithms", Theoretical Computer Science 259, 1-61,2001.

[9] M. Boughanem, C. Chrisment, L. Tamine, "Multiple query evaluation based on an enhanced genetic algorithm", Information Processing and Management 39,215-231,2003

2. Sun Kim and Byoung, Genetic Mining of HTML structures for effective information retrieval , Applied Intelligence , 18, 243256,2003

[10] H. Chen, C. Yi-Ming, M. Ramsey, C. Yang, "An intelligent personal spider (agent) for dynamic Internet/Intranet searching", Decision Support Systems 23 (1998) 41-58.

[11] S. M. Khalessizadeh, R. Zaefarian, S.H. Nasseri, and E. Ardil , Genetic Mining, Using Genetic Algorithm for Topic based on Concept Distribution, PROCEEDINGS OF World academy of science, engineering and technology ,143 -147, 2006

[12] Suhail S.J Komer, Dsan Husek, Using genetic algorithms for Boolean optimization, Proceedings of Conference, IEEE,178-183

[13] D. Vrajitoru, Crossover improvement for the genetic algorithm in information retrieval, Information Processing and Management 34 (4) ,405-415 ,1998

[14] O.Cordon, .E.Herrera , C. Lopez- Pujalte, M.Luque, C.Zarco ,A review on the application of evolutionary computation to information retrieval, International Journal of Approximate reasoning34, 241- 264, 2003.

[15] Roci' 0 L. Cecchini , Carlos M. Lorenzetti , Ana G. Maguitman, Nelida Beatnz Brignole, Using genetic algorithms to evolve a population of topical queries , Information Processing and Management, 2008

[16] A.M. Robertson, P. Willet, Generation of equifrequent groups of words using a genetic algorithm, Journal of Documentation 50 (3) ,213-232., 1994

b. P. Pathak, M. Gordon, W. Fan, Effective information retrieval using genetic algorithms based matching functions adaptation, in, Proc. 33rd Hawaii International Conference on Science (HICS), Hawaii, USA, 2000.

SP0913008