NOTE TO USERS

Page(s) not included in the original manuscript and are unavailable from the author or university. The manuscript was scanned as received.

74-79

This reproduction is the best copy available.

UMI®

uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Quoc-Tuan Nguyen

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.Sc. (Systems Science)

GRADE / DEGREE

Systems Science

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Using the Genetic Algorithm to Optimize Web Search: Lessons from Biology

TITRE DE LA THÈSE / TITLE OF THESIS

Evanne Casson

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

May Griffith

Stan Matwin

Gary W. Slater

LE DOYEN DE LA FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES /
DEAN OF THE FACULTY OF GRADUATE AND POSTDOCORAL STUDIES

USING THE GENETIC ALGORITHM TO OPTIMIZE WEB SEARCH: LESSONS FROM BIOLOGY

Nguyen Quoc Tuan

Thesis Director

Evanne Casson, Ph.D.

A thesis submitted to the Faculty of Graduate and Postdoctoral Studies
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE
SYSTEMS SCIENCE

University of Ottawa
Ottawa, Canada

Your file Votre référence
ISBN: 0-494-14933-7
Our file Notre référence
ISBN: 0-494-14933-7

ABSTRACT

Searching for information on the Web is a relatively inefficient process. It is time consuming and requires a learning process. Modern search engines work by "crawling" the Web, scanning web pages and indexing them at a central repository. As the number of web pages on the Internet grows exponentially, so does the size of the indexes, and the number of results returned by a query. For most search engines, a user must manually refine the query by changing the keywords to eliminate irrelevant results and to bring the most relevant results to the top of the search engine output. My goal is to develop a method that optimizes web search queries without user intervention. Developing intelligent ways to automate this process includes the development of algorithms that automatically manipulate the use of keywords to produce the desired output. Genetic algorithms (GA) provide a potentially useful approach in this area. However, these approaches have not fully exploited the biological concepts associated with genetic reproduction and evolution. I hypothesize that an approach that uses GA but modifies it to include the biological concepts of structural and regulatory gene types and the use of a combination of deletion operator and silent genes will improve GA performance in optimizing Web search. In this paper, I describe this approach and its implementation in simulations of Web search tasks using three popular Web search engines (Google, Yahoo and Netscape). The results of this implementation are presented and are compared to the performance of a similar, but unmodified GA in the same tasks. The comparisons demonstrate that, in almost all conditions, the GA modified to include the representation of both structural and regulatory genes performs better than the unmodified GA in terms of the average fitness value of the final search results, although this conclusion depends to some extent on the nature of the search task and search engine. As expected, the addition of a deletion operator has the most effect on the average fitness when silent genes are present. The results of the deletion operator and silent gene concept shows the potential of the modified GA approach for situations where the key words entered by users are less relevant to the search engine than the user intended.

ACKNOWLEDGEMENTS

There are no words to express my deepest gratitude to Dr. Evanne Casson, my supervisor. Without her, I would not have been able to finish this thesis work. She was there for me, to encourage me, and to help me believe in myself. Through her, I found enjoyment and satisfaction with my thesis work.

I would like to express my gratitude and love to all members of my family and my uncle Huong's family for their support; especially my brother Tuyen Nguyen, who took over my building management responsibilities and worked many late nights and weekends in addition to his main job, so I would have more time to finish my thesis. A special thanks also to my brother Tu Nguyen for his valuable feedback on my software work as well as his dedicate support.

I am indebted to my work colleagues and friends, Mike Fissell and Gerry Libretti. Even with their very heavy workload, they found the time to read my thesis and provided me with valuable commercial points of view.

I am grateful for the feedback of Dr. Duc-Long Nguyen for his non-computer science perspective of my thesis work, for taking time to read my work: Belinda Lam, Ramin Dowlati, Sansan Lee, Tom Williams, and for the support of Nam La and Bich Ta. My special thanks to my dear friends Kim-Ngan Trinh, Denise Su and Thu-Van Nguyen for their support.

I also would like to thank Dr. Stan Matwin from University of Ottawa, Canada for his advice and Dr. H.C. Paul Lee from National Central University, Taiwan for listening to my ideas and for providing guidance.

THANK YOU!

TABLE OF CONTENTS

ABSTRACT iii
ACKNOWLEDGEMENTS iv
TABLE OF CONTENTS v
LIST OF TABLES vii
LIST OF ILLUSTRATIONS viii
CHAPTER 1: INTRODUCTION 1
Overview of Web Search 1
Overview of Approaches to Optimizing Search Queries 7
Overview of Basic Genetic Concepts 9
Genetic Algorithms and Web Search 11
GA and Web Search 14
Modifying GA to Optimize Web Search Engine Queries 16
Hypothesis 20
CHAPTER 2: PROGRAMMING GA 22
Representation 24
Genetic Algorithm 27
Crossover: *29*
Mutation: *30*
Mapping Genotype to Phenotype: *31*
Selection: *31*
Genetic Algorithm Parameters 32
Genetic Algorithm Output 33
Fitness 33
CHAPTER 3: TESTING GA 36
Real Time Implementation 38
Implementation by Simulation: 39
Implementing the Fitness Database 42
Implementing the Search Engine Subsystem 43
Implementing the Automatic Ranking Subsystem 45
Test Cases 46
CHAPTER 4: RESULTS 48
Performance of Simulation Sub-systems 48
Performance of GA Simulation 49
Impact of Simulation Parameters 52
The Effect of the Number of Simulations *52*
Effect of Number of Generations: *55*
Effect of Initial Population Size: *57*
Comparison of GA Performance 59
Structural GA: *61*
Regulatory GA: *61*
Deletion GA: *61*
Cases: *62*
Search Engines: *62*
DISCUSSION 64

Summary of Results 64
Performance of Simulation Subsystems: *64*
Performance of Simulation Parameters: *64*
Comparison of GA Performance *64*
Simulation Subsystems and Parameters: *65*
Comparing the Performance of Structural, Regulatory and Deletion versions of GA . 66
Impact of Modified GA: *66*
Impact of Cases: *66*
Impact of Search Engines: *66*
Limitations of this research: *67*
Reverse Engineering: *68*
Impact on Web Search: *69*
CONCLUSION 70
REFERENCES 71
APPENDIX A 80

LIST OF TABLES

Table 1: Comparison of concepts *13*
Table 2: A comparison of the current and modified GA approaches for optimizing Web search engine queries. *17*
Table 3: GA summary *21*
Table 4: Mapping of GA terms to Web Search terms *24*
Table 5 : Summary of GA sub-processes for both the Current and Modified GA program modules *32*
Table 6 : GA Parameters *33*
Table 7 Real time implementation of test execution *39*
Table 8: Non real time implementation of test execution in simulation mode *42*
Table 9: Example of how fitness values are stored in database for <purchase, agreement, of> *42*
Table 10: Outline of issues relating to the development of a programmatic interface to web sites along with the solutions adopting in the coding *43*
Table 11: Outline of process by which Search Engine subsystem collects output web pages *44*
Table 12: Results of GA parameters choices *52*
Table 13 Statistical Analysis of Results using 1600 Simulations with Generations =12 and Initial Population Size = 20 ..... *60*

LIST OF ILLUSTRATIONS

Figure 1: Example of Yahoo Hierarchical Categories 2
Figure 2: Example of Google search based on keywords 3
Figure 3: Example of Google search based on keywords and search operators 4
Figure 4: Flow diagram of the general approach used by spider-based search engines 5
Figure 5: Left panel shows basic IR system while right panel shows IR system with index as used by Gordon (1988). 14
Figure 6: Yang and Korfhage (1993) added user feedback to basic IR system 15
Figure 7: Modified GA approach versus unmodified GA approach 18
Figure 8: Diagram showing how the concepts of silent gene/deletion can be applied to Web search as a nonsense keyword and deletion operator 19
Figure 9: Overview of Genetic Algorithm components: A genetic representation that maps phenotype (solution domain) into genotype (genetic domain), a genetic algorithm process that evolves genotype based on natural selection plus applications of genetic operators such as crossover and mutation, a mapping process that maps genotype back to phenotype, and a fitness determination process that ranks the phenotypes for the purpose to be included in the next generation. 23
Figure 10: Current representation example of GA (called Structural in this study) 25
Figure 11: Modified representation example of GA (used in both Regulatory and Deletion GA <change from quotation to and>) 26
Figure 12: Interaction between structural and regulatory genes to create genotype in Modified GA representation 27
Figure 13: Implementation of GA 28
Figure 14: Ranking mechanism 35
Figure 15: GA real-time implementation in detail 37
Figure 16: Real time implementation of GA: Overall Architecture 38
Figure 17: GA simulation implementation in detail 41
Figure 18: A sample of Java code translated from ranking rule 46
Figure 19: Third experiment using keywords: Hotel, Place, Des, Arts. 47
Figure 20: Example of results for individual simulations of the Regulatory GA condition (left panel) showing 10 simulations of Google-Case 1. A summary of these results is presented (right panel) with average fitness values and standard deviations for each generation. 51
Figure 21: The effect of number of simulations for all cases (columns) and search engines (rows). Results for all three GA conditions shown in each graph. 54
Figure 22: The effect of number of generations for 2 examples: Case 1 with Google (left panel) and Case 3 56
Figure 23: The effect of initial population size for 2 examples: Case 1 with Google (left panel) and Case 3 with Netscape (right panel). The other parameters are generations=20, simulation runs=1600. The three GA conditions are plotted in each graph, showing the average fitness and standard deviation 58
Figure 24: GA performance across all cases (columns) and search engines (rows). Average fitness for the three types of GA shown in each panel along with standard error bars. Parameters are: initial population=20, generations=12 and simulations=1600 63

CHAPTER 1: INTRODUCTION

Searching for information on the Web can be time-consuming and requires a learning process. Search engines, which were developed to facilitate the process of searching the Internet, are essentially text retrieval systems. They provide the user with enhanced search capabilities by "crawling" the Web, scanning web pages and indexing them at a central repository, which can then be accessed using a query containing key words. However, as the number of web pages on the Internet grows exponentially, both the size of the indexes and the number of results returned by a query also increases. For most search engines, the user must manually refine the query to eliminate irrelevant results by changing the keywords. Updating a query to adapt to the rapidly changing Web environment also requires manual user intervention. Automating these processes requires developing intelligent search engines and query processes that can refine the search process to make it more focused, yet remain flexible enough to respond to the rapid changes in information on the Web.

Overview of Web Search

The goal of most simple search engines is to locate specific information on the Web that matches keywords supplied by a user. Search engines try to return the most accurate results possible, as fast as possible. In this paper, I would like to focus only on large-scale hyper-textual Web search engines; thus, the term "search engine" will be used to mean "web search engine". There are two basic types of search provided by most search engines: searching by keywords and browsing a category hierarchy.

In browsing by category, the search space is organized by category in a hierarchy. For example, Figure 1 shows one can find information about a Real Estate Purchase Agreement in Ontario using Yahoo Hierarchy Search.

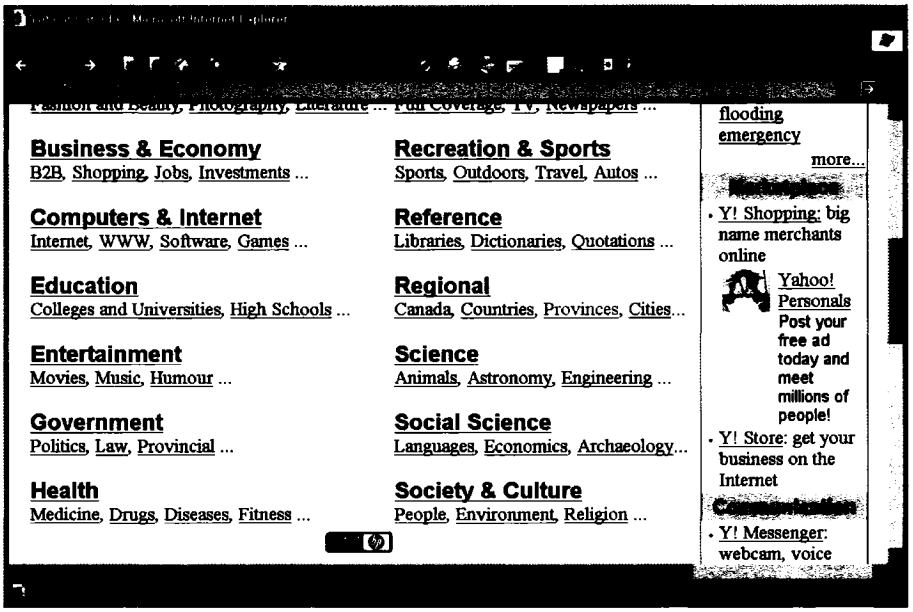

Figure 1: Example of Yahoo Hierarchical Categories

Using this approach, the search process can drill down from category Canada to category Province, until the category Real Estate is reached. From here, a legal text of an agreement to buy a house can be found. Searching based on category requires the user to spend a lot of time to drill down the search space to find information. The success and efficiency of the process depends on the number of categories and sub-categories. If there are many branches or paths for a category or sub-category, it may be necessary to browse them all, making the search process time-consuming. On the other hand, if there are a limited number of categories the search will be more rapid, but there may only be a limited amount of information available, making the search less relevant.

The other common method used by the majority of search engines is to accept keywords entered by the users and to return all web pages containing the information that users are looking for. Google is an example of a search engine that functions in this way. Google maintains and updates a cache of web pages and an index of keywords. The user enters keywords in the form of a simple query, which is processed by the Google search engine. Google then returns a list of related hyperlinked web pages (URLs) and their brief descriptions. Browsing brief descriptions, the user may choose to examine the contents

of one or more web pages, which are thought to potentially contain the desired information. Usually, the user will examine, at most, the top twenty or thirty returned web pages, while the number of returned web pages could range from a few hundred to thousands and even to millions. If the desired information does not appear in the searched web pages, the user might try again adding search operators to the keywords to ensure that the returned web pages contain all searched keywords.

For example, Figure 2 shows how keywords are normally entered into search engine and Figure 3 shows how the search operator "AND" or symbol plus "+" can be used to ensure the occurrences of certain keywords.

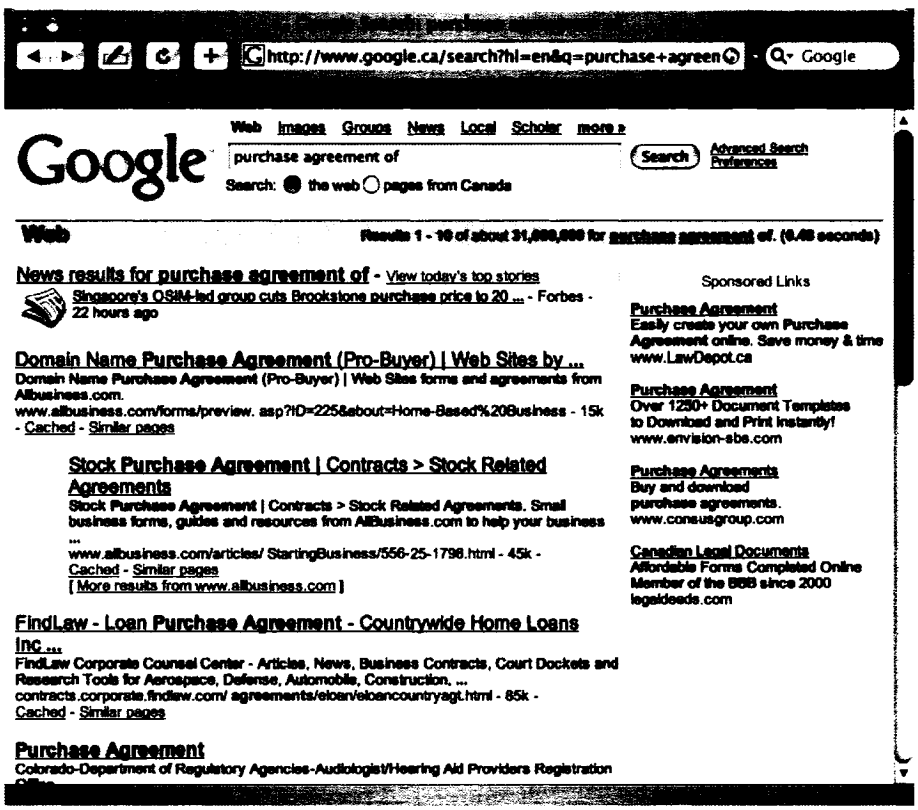

Figure 2: Example of Google search based on keywords

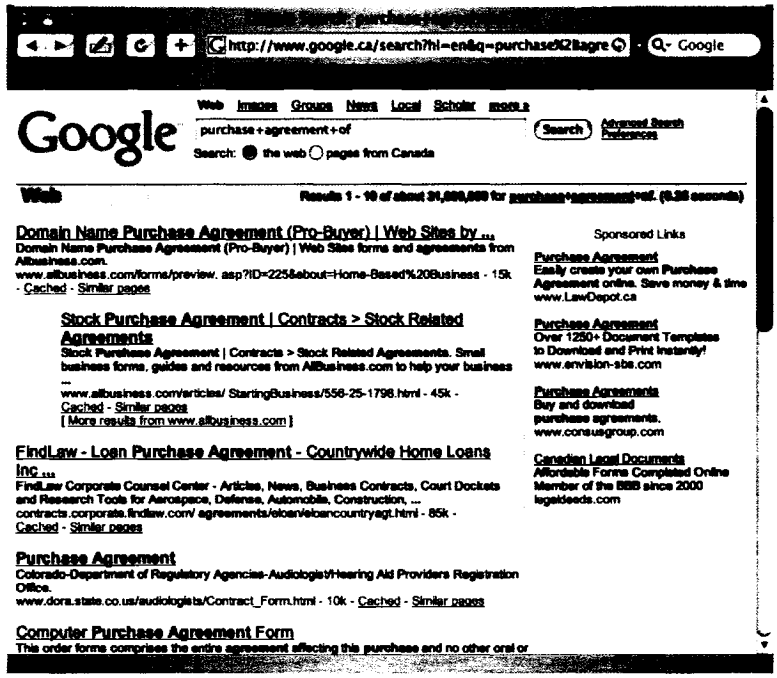

Figure 3: Example of Google search based on keywords and search operators

The user can manipulate keywords and search operators in different patterns in the hope of reducing the number of returned web pages, and to try to get the desired information in the top ten hyperlinked web pages (URL listings). One of the common techniques is to combine keywords into a sub-phrase in certain orders. The pattern of keywords in a particular sequence might efficiently reduce the number of returned web pages. However, with multiple keywords, there are so many combinations such that users might not manually try all of them. This trial and error process is repeated until the user is satisfied with the results or abandons the search. In addition to other operators, the deletion operator, which is marked with symbol <*space*>-, can be used to exclude certain keywords from returned web pages.

The efficiency of the search and the query process depends on the type of search engine. There are only a few basic types of search engine architectures: spider based search engines and distributed search engines.

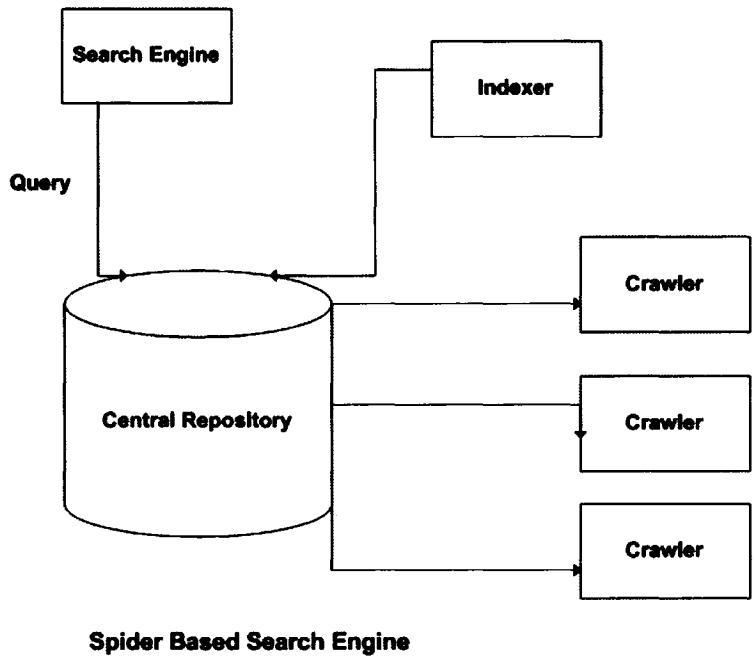

Figure 4: Flow diagram of the general approach used by spider-based search engines

As Figure 4 demonstrates, spider-based search engine architectures mainly consist of three components: crawlers, repositories, and indexers[1].

Crawlers wander on the "Net", collect documents and send them back to the central server. At the central server, documents are stored in a repository. The indexer performs indexing functions on them, mapping keywords to documents. Different search engines use different techniques to index as well as different index types and frequencies of indexing. Most of them extract all the "links" in each web page and store important link information such as where the link came from, where the link goes to and the text description of the link.

Google, Lycos, InfoSeek, Yahoo, AltaVista, AOL and Excite belong to the Crawler search engine category. However, Google was the first to develop new methods for more efficient indexing. It introduced a different way of breaking down a very large index into smaller ones and it distributes the workload of the indexer and page ranker among

hundreds or thousands of smaller computers. This innovation differentiated Google from other search engines in terms of the speed of the query response. The query response from Google is very fast. Over time, Yahoo search performance has caught up with Google's search performance[2]. With the purchase of Overture, which had acquired Alta Vista, Yahoo also inherited Alta Vista technology. As it is right now, Yahoo crawling search technology is a combination of Alta Vista, All-The-Web and Inktomi. This crawler-based technology, which originated from UC Berkeley, was selected by Yahoo to provide faster search capabilities. Netscape, on the other hand, unlike Yahoo, uses Google as its main listing, but puts its own paid-listing content at the top of the search results.

Distributed search engines are designed to have many databases of documents and use peer-to-peer networking concepts. Examples are Napster technology (not Napster itself)[3] and Gnutella[4], which are quite efficient in allowing users to search and exchange music files. While Napster still relies on centralized servers for global indexes and searching, other systems such as Gnutella are completely decentralized[5]. The main theme of the peer-to-peer concept is that there is no central server and each node has it own database and acts as both client and server. This type of search engine is still limited to specific tasks such as downloading music and movies. Although this new concept could be very powerful for other kinds of web based information searches, to date, the type of keywords that can be used, and files that can be searched for are still quite restricted.

In summary, the problem with a keyword search engine such as Google is that, while it is simple and fast, from the user's perspective, it is not always efficient. An increasingly overwhelming number of web pages come back from most user queries and the task of manually reformatting the query to reduce irrelevant web page references and make sure the relevant web pages appear at or near the top of the listing is time consuming. Furthermore, despite the large numbers of results that come up from many queries, no search engine is yet available that is able to get all the relevant information on the Web consistently.

Most search engines attempt to improve their performance by increasing the relevance of the result. This is done using various ranking algorithms together with sophisticated indexing mechanisms. In addition to improving the indexing of relevant information, search performance can be improved by improving the way that irrelevant information is eliminated. However, in some cases, this can cause relevant web pages to be incorrectly eliminated. In all cases, these improvements to the search process require that the search engine itself to be modified.

An alternative approach to improving search performance, which does not involve modifying the search engine, focuses on optimizing query inputs to search engines. With this approach, algorithms could be developed that can manipulate the use of keywords to refine the search process to reduce irrelevant results. Methods to optimize queries rather than search engines are more likely to be useful in the current situation where a small number of proprietary search engines have come to dominate the Internet.

Overview of Approaches to Optimizing Search Queries

Methods for optimizing search queries have been developed in the area of information retrieval (IR). Web search engines represent a form of IR system. IR systems have three main components: (1) a database of documents, (2) query processing and (3) pattern matching[6]. Documents are indexed in a database using an indexer. Queries submitted by users are processed into the appropriate form for pattern matching. Relevant documents are then retrieved from the database and evaluated by the pattern matching subsystem. This subsystem will see if retrieved documents have the satisfactory degree of relevancy or not. If they do not, then the cycle of retrieving and evaluating stored documents will repeat until satisfactory documents are obtained. The final documents will be returned to users. There are many different models to represent the IR system. They are Boolean model, Vector Space model, Probabilistic model and Fuzzy model. Some of these models differ in how they perform indexing. In the Boolean model, the core of the database is the binary index, where value of one (1) means key word appears at least once in the document; otherwise, the value is zero (0). In the Vector Space model, the document is viewed as a vector of n-dimensions where n is the number of key words

associated with the document. A query is also a vector of n-dimension where n is the number of key words. There is a weight value associated with the key words in the query. The relevancy degree is derived from the similarity comparison between query vector and document vectors. Other models such as Probabilistic model and Fuzzy model use search functions that are based on different probability techniques[6]. The probabilistic model uses distribution of indexes to implement the search function in query processing. The Fuzzy model defines a fuzzy mapping, between a document and a set of documents representing the query, which is used as index function.

From the above review, it is clear that current search engines, irrespective of different approaches and optimizations, can benefit from a more effective query method. Several attempts have been made to make the query process more effective for individual users. These methods have used a number of different ways to improve the query process, and some try to improve the search engine itself. Since Google, Yahoo, AOL and Microsoft now dominate the search service in the Internet, it is likely difficult to propose changes to search engine architecture or algorithms. Furthermore, most methods are not designed to provide dynamic solutions for an immense and constantly changing search space such as the Web. However, both the user requirements and the information on the Web are dynamic processes. Thus, a truly effective process to refine and customize the results of searching the Web to an individual's requirements should also be dynamic.

Dynamic solutions require a different approach; an approach that can evolve the queries with the user's needs and changes in the available information. Machine learning approaches such as genetic algorithm, genetic programming, neural network and decision tree methods are examples of approaches that may be used [7]. At this time, there is little evidence[8] to suggest which of these approaches will work best at optimizing query for web search[9]. For this thesis work, I have selected the Genetic Algorithm for the following reasons. First of all, this approach has been found to be useful where the problem domain is very large and complex[10]. Secondly, this approach is effective for a dynamic problem domain such as the Web[11]. Thirdly, the Genetic Algorithm is more suitable for a problem domain such as searching on the web where there is no single

optimal solution. Finally, this approach will allow me to incorporate additional biological concepts. This is important, as the main goal in this research is to evaluate the impact of additional biological concepts in developing a solution for this problem.

The Genetic Algorithm (GA) was developed by Holland (1975)[12]. The GA approach to machine learning is based on the principles of evolution and genetics. Before discussing the way that GA works, it is important to describe some of the basic biological concepts involved.

Overview of Basic Genetic Concepts

Natural selection and the genetic code are the basis of the biological algorithm of directed change that we call evolution. The genetic code is encoded by genes, which are composed of blocks of Deoxyribonucleic acid, or DNA. The pattern of DNA within each gene provides the information necessary to build a protein; the basic building block of the organism. Genes are found in supra-molecular structures known as chromosomes. The position of the gene on the chromosome is called the locus. Different forms of the same gene, encoding different forms of the protein, are called alleles. A complete set of chromosomes, which include all the genetic material, is called a genome. A genome is often defined as the entire genetic information of an organism while a gene is a unit of genetic information. A particular set of genes or alleles in a genome represents a genotype. In that development, some genes, called structural genes, are involved in directing the production of individual proteins, while other genes, called regulatory genes, act by turning on (activating) or turning off (repressing) the activity of particular structural genes.[13]

During the process of sexual reproduction, the genetic material of the parents is combined by a process called "crossover" to form a new chromosome. As the genetic material of the parents chromosomes (diploids) split to form four different sets of genes (haploids) and then recombine, the new generation is formed from random combinations of genetic material from both parents allowing new genotypes and, once the organism develops, new phenotypes to arise.

In addition, the genes contributed from each parent can undergo a process called mutation where the sequence of genes can be altered; creating new genetic material that was not present in either parent. A gene mutation is a case where an allele of a gene changes to a different allele. This will result in a different protein being produced or cessation of protein production and a different phenotype. A point mutation is a small gene mutation: for example, a change in the code of a single amino acid. Some mutations occur in parts of the genetic material that do not appear to have an effect on the phenotype (silent genes). These mutations are called "silent mutations"[a]. In addition to the different types of mutations described above, the mutation consists of number different operations: deletion, insertion, duplication, and inversion.

Evolution depends on the concept of variation in genotype and phenotype among individuals within a population. As natural selection takes place, some individuals are more successful than others in surviving and in reproducing; that is, they have a higher fitness. As a result, their genetic material is more likely to be passed on to a subsequent generation in combination with the genetic information of the other parent. Similarly, the genetic material of individuals with less fitness is less likely to be reproduced and will be less represented in subsequent generations. Over generations, the population gradually contains more fit individuals and fewer unfit ones. The process continues in this manner until the environment changes in some way, changing the process of natural selection and the phenotypic characteristics that are desirable. The definition of fitness changes and the evolution goes in another direction.

The ability of a population to evolve depends on having enough variation in genotype and phenotype in the initial population, having a process of natural selection (that is, some way of determining fitness) and by having enough generations.

Since they occur randomly, generally mutations do not generally produce viable phenotypes. But they are an important part of the process of evolution as they provide

---

[a] Definition taken from http://en.wikipedia.org/wiki/Silent_gene

the possibility of new genetic solutions that may produce fitter phenotypes in a continuously changing natural environment.

Genetic Algorithms and Web Search

It has been suggested by a number of authors the concepts adapted from the biological algorithm can be used to solve complex problems[9]. Rechenberg initiated Evolution Strategies to use the principle of evolution to solve parameter-optimization problems. John Holland, who helped to make the concept of the Genetic Algorithm well known, emphasized the potential of solving complex problems using biological concepts of natural selection and genetic inheritance. In this approach, potential solutions to a problem are represented as a genotype, with the solution product as the phenotype. The population of individual solutions is subject to such genetic operators as reproduction, crossover and mutation, producing new solutions and solution products. Each solution and its associated product have a fitness value associated with it. In each generation, individuals breed and reproduce with a probability proportional to the value assigned to it by a "fitness" function. Unfit solutions become a smaller and smaller part of the gene pool. Eventually, the process converges on an optimal solution with a specific genotype acquiring a dominant position in the population.

Unlike other machine learning approaches, the basic genetic algorithm does not use a training data set. Training data set have been used with GA in hybrid approaches[14]; however, the use of training data set is not appropriated in this problem domain due to the uniqueness of the solution for each user and set of key words.

This approach can be applied to the problem of optimizing Web search. Queries can be represented by the genotype, with keywords and operators as individual components of the genotype. In this case, phenotype, or product of the genotype, will be the individual search results from a specific query. Fitness of a specific genotype can be determined by evaluating the relevance of the search result, similar to the ranking method used in several of the approaches described above.

If the algorithm functions correctly through successive generations, the majority of resulting searches (population) would become increasingly relevant and efficient for the specific needs of the user and the current state of the Web. However, just as interesting is the fact that the population will continue to contain other genotypes and express diverse solutions with a lower probability of reproducing. While those genotypes would not be immediately relevant, they hold the potential for an evolving solution that can meet future requirements. Furthermore, this pool of genetic diversity ensures that if a "wrong turn" is taken in the evolutionary process, no relevant information is lost, as all genetic information remains represented in the gene pool. Thus, premature terminations in the search process are not as likely to occur.

These concepts are summarized in Table 1 to allow an easy comparison of how biological concepts can be applied in a general genetic algorithm and more specifically to a Web search problem.

Table 1: Comparison of concepts

Biological Concept	**Genetic Algorithm**	**Web Search Problem**
Genome	Solution search space	All possible queries
Gene: A gene is a block of DNA. Each gene encodes a particular protein	Gene: Each gene encodes a part of a solution	Gene: Each gene is either keyword or search operator
Chromosome: A chromosome consists of genes	Chromosome: general solution protocol	Chromosome: query format: keywords and operators
Genotype: Blue print to construct organism, the DNA	Genotype: Solution	Genotype: A query input into search engine
Phenotype: Physical organism	Phenotype: Implemented solution	Phenotype: Search result
Fitness: Ability to survive and reproduce	Fitness: Value of solution	Fitness: User satisfaction with the output web pages
Mutation: Mutation alters one or more genes on a selected chromosome	Mutation: Alters solution to produce new solution	Mutation: Change keywords to explore an optimized query
Crossover: method of genetic transfer from parents to offspring	Crossover: Creates new combinations from solution set	Crossover: Change sub-phrase or a subset of keywords/search operator to exploit an optimized query
Evolution: Directed change in response to environment over generations	Evolution: Repeated iterations of GA until an optimum solution is found or a condition is satisfied	Evolution: Try different query inputs until user is satisfied

GA and Web Search

The idea of using GA for IR is not new. Gordon (1988)[1516] used GA in an effort to solve the IR problem of improving the way web page documents are indexed (see Figure *5*). He used keywords associated with a particular web page document as a gene in GA binary encoding. These bit strings could be modified by mutation to come up with a different set of keywords. Each web page document had multiple descriptions or keywords associated with it. A Jaccard[17] score function was used to implement the GA fitness function. The average of those scores was used to determine if the document should be retrieved or not.

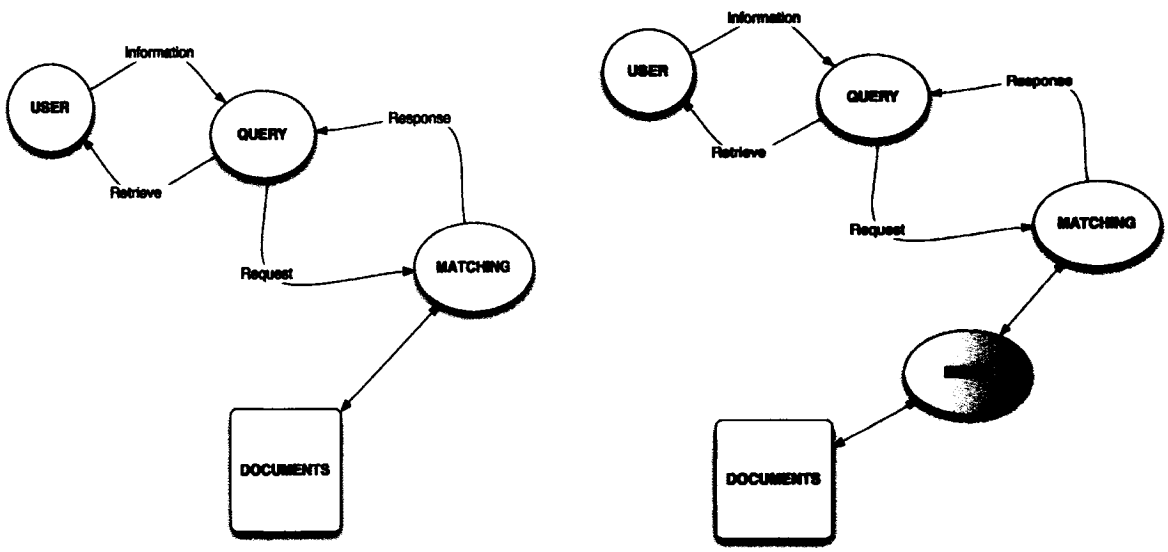

Figure 5: Left panel shows basic IR system while right panel shows IR system with index as used by Gordon (1988).

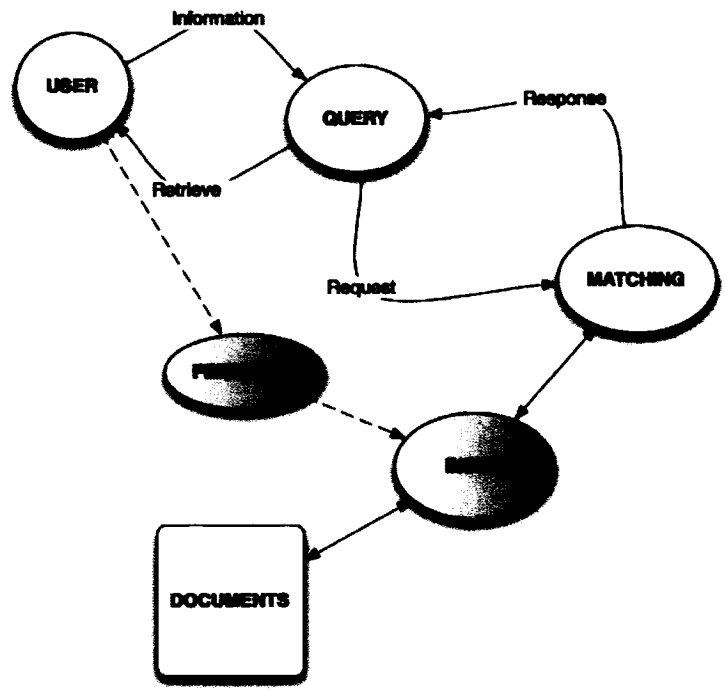

Figure 6: Yang and Korfhage (1993) added user feedback to basic IR system

Yang and Korfhage (1993)[18] also used GA to develop a better method to index web page documents. Different from the Gordon approach, they weighted documents based on the feedback of user ranking, they use random number generator to obtain probability for mutation, and two point crossovers. With user feed back, the weight or rank of query term is more accurate. Hence, the fitness function better determines the fitness (See Figure 6).

In summary, Gordon, Yang, Korfhage and most of the GA work on information retrieval have focused on how to improve the search engine performance by improving the indexing of web page documents, rather than extracting information from the index as occurs in the search process.

The approach described here is quite similar to that of Gordon and Yang and Korfhage in that GA is being used as a way to optimize the search engine performance. However, it is also quite different in that I am focusing on optimizing the query process by

manipulating input keywords and using Web-based search engines. These are treated as black boxes, which accept input to, and produce output for, the GA. As pointed out above, optimizing the query process has the potential to benefit all search engines. The use of GA to solve this problem will also allow more flexibility and dynamic response by improving the searches. Keyword optimization also could be a helpful as it can be used to improve our understanding of search engine behaviors as well as how proper keywords can be used to lead to the higher ranking of a web site.[19]

Although the work described above demonstrates the feasibility of a GA approach, the representation used in most examples of GA may not provide the best model for representing a query. A complete query consists of both keywords and operators, which have different functions within the query. Since these two variables have different properties and are drawn from different language populations, it is likely that they should be represented differently in the GA. This type of heterogeneous representation is actually more equivalent to the original biological model, in which some genes are structural while others are regulatory (see above). An approach similar to this has been suggested but involves more complex concepts from molecular biology[20]. I have developed a slightly different GA model based more closely on the biological model to allow for the incorporation of features that are more representative of the problem of refining search queries to provide more effective searches. In addition, to make queries more dynamic, it may be useful to incorporate an additional mechanism that ensures the responsiveness of the biological system: the concept of silent gene and gene deletion.

Modifying GA to Optimize Web Search Engine Queries: In modifying the GA representation to include more biological concepts, I focus on two specific enhancements. The first enhancement is to modify the structure of chromosome such that it composed of more than just one type of gene. The second, closely related enhancement, is to apply the concept of gene deletion and something similar to the concept of a silent gene; that is a gene that does not have much impact on the genotype. In Web search, the concept of a silent gene is similar to the use of keyword that does not make sense: a nonsense keyword. Table 2 shows how these concepts can be applied to modify the GA. In the

Table 2, the differences between the unmodified GA and the modified GA as applied to the problem of query optimization in Web search are set out.

Table 2: A comparison of the current and modified GA approaches for optimizing Web search engine queries.

Current GA Approach	Modified GA Approach
Homogeneous Representation of Genes: The basic GA defines chromosome **as a set of genes from the same type, that is, the representation of keywords is homogeneous.**	Heterogeneous Representation of Genes: The basic GA definition for chromosome **is modified to contain both regulatory genes and structural genes; that is, keywords are represented differently than operators.**
Deletion: **No deletion operation**	Deletion: **Deletion implemented as an additional operator in representation**
Silent Gene: **No silent gene** representation	Silent Gene: **Silent gene implemented as additional "nonsense" keyword**
Genetic Algorithm: GA, parameters and execution are the same in both implementations	Genetic Algorithm: GA, parameters and execution are the same in both implementations
Genotype to Phenotype Mapping: Mapping from genotype (query) to phenotype (search results) is the same for both implementations	Genotype to Phenotype Mapping: Mapping from genotype (query) to phenotype (search results) the same for both implementations
Fitness: The fitness procedure is the same for both implementations.	Fitness: The fitness procedure is the same for both implementations.

Figure 7 also shows how these two approaches differ. In one path (current GA) the representation is modeled as homogenous and the other (modified GA) is heterogeneous. My hypothesis is that the modified GA approach will perform better than the unmodified GA approach in optimizing the queries to produce the search results desired by the user. This improved performance is independent of the choices of values for GA parameters such as mutation, crossover, selection as well as the fitness function.

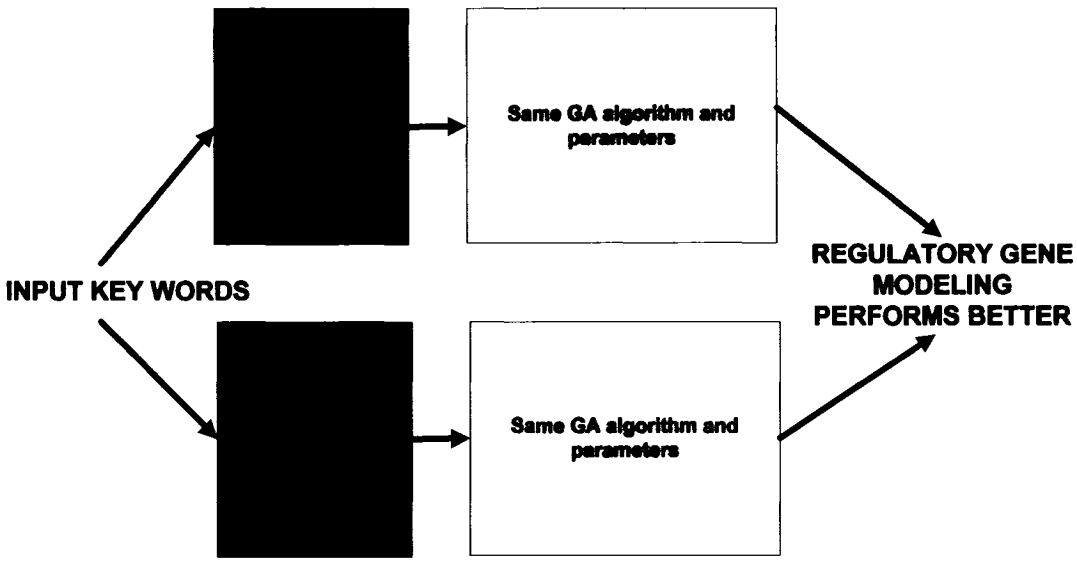

Figure 7: Modified GA approach versus unmodified GA approach. Consider any Genetic Algorithm implementation as black box, a regulatory and structural genes model performs better than homogenous genes model.

In addition to the concept of heterogeneous representation of genes, I hypothesize that incorporating the concept of gene deletion used in combination with silent genes into the GA will also improve its performance in optimizing Web search. These concepts represent important biological tools for altering the genetic code producing new solutions. A silent gene, when combined with mutation, represents the potential for surviving a change in evolutionary direction. A mutation in this gene may provide a "good" gene in a changing natural environment. However, a mutation of a silent gene is more likely equal to be a less fit ("bad"). Therefore, it is useful to have some kind of DELETION operator that might remove these "bad" genes. Figure 8 demonstrates this concept graphically.

In terms of biological concepts, DELETION would be one of the mutating processes involved in the GA algorithm. However, it is much easier for the purpose of application of biological concepts to search engine optimization, to incorporate DELETION as a keyword operator in representation; that is, to treat it as a regulatory gene in the model. Similarly, to maintain the integrity of the GA algorithm as a black box, I have incorporated the biological concept of silent gene into the representation as a nonsense keyword, which is a structural gene.

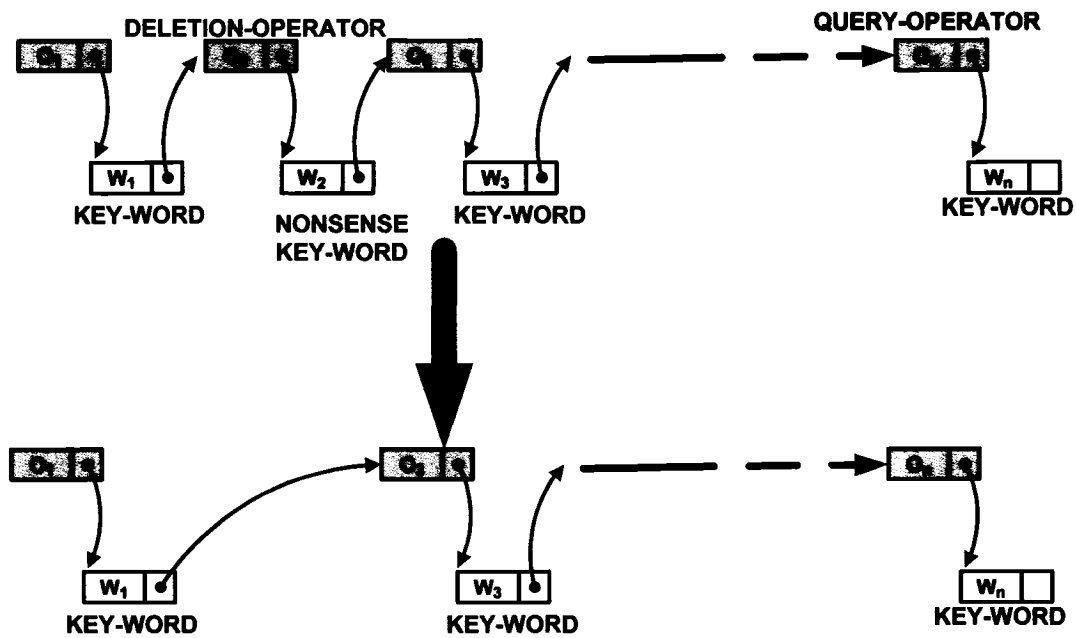

Figure 8: Diagram showing how the concepts of silent gene/deletion can be applied to Web search as a nonsense keyword and deletion operator.

Hypothesis

My hypothesis is that incorporating the concept of heterogeneous representation of genes as well as combining silent genes with deletion operators in an implementation of GA designed to optimize the query input to Web search engines will improve the effectiveness of the query and produce more relevant information in the top (first) output pages. The improvement may depend on the type of search engine; search engines that are similar, such as Google and Netscape are expected to produce similar increases in effectiveness, while search engines based on different technology may produce different results.

To test this hypothesis, the performances of current and modified GA implementations are compared. Three different GA representations are implemented to allow a comparison of the different concepts discussed above: (1) GA with homogenous representation of structural genes only (this will be referred to as the Structural GA), (2) GA with heterogeneous representation of genes, composed of structural genes and regulatory genes (this will be referred to as the Regulatory GA) and, (3) GA with heterogeneous representation of genes with the addition of a deletion operator as another kind of regulatory gene (this will be referred to as the Deletion GA). These GA conditions will be tested against three test cases: Case 1, consisting of common words combined to search for information on a specific item "purchase, agreement, of"; Case 2 where the concept of a silent gene appears , with the addition of a nonsense key word, "smith" to the keywords in Case 1; Case 3, which uses a place name to search of information about travel "hotel, place, des, arts". These test cases will be executed against three different search engines: Google, Yahoo and Netscape.

Table 3 shows the terminology that will be used to describe the implementation and results along with the type of GA.

Table 3: GA summary

Terminology	**Structural GA**	**Regulatory GA**	**Deletion GA**
	Homogenous representation	Heterogeneous representation	Heterogeneous representation
	Unmodified GA	Modified GA without deletion	Modified GA with deletion

CHAPTER 2: PROGRAMMING GA

The process of modifying GA representation to improve the results of GA optimization of search engine queries involves two basic components: developing the GA programs (Programming GA), and then developing the method of assessing their performance (Testing GA). Due to problems with rapid, iterative access of Web-based search engines, executing the GA on-line was not possible. Therefore, the section on Testing GA describes an off-line simulation of this process.

As Figure 9 demonstrates, implementation of GA for the specific problem of optimizing web search can be divided into four logical steps: (1) representation of genotype input to the GA, (2) the implementation of GA to produce new genotypes, (3) the mapping from genotype to phenotype, and (4) fitness determination.

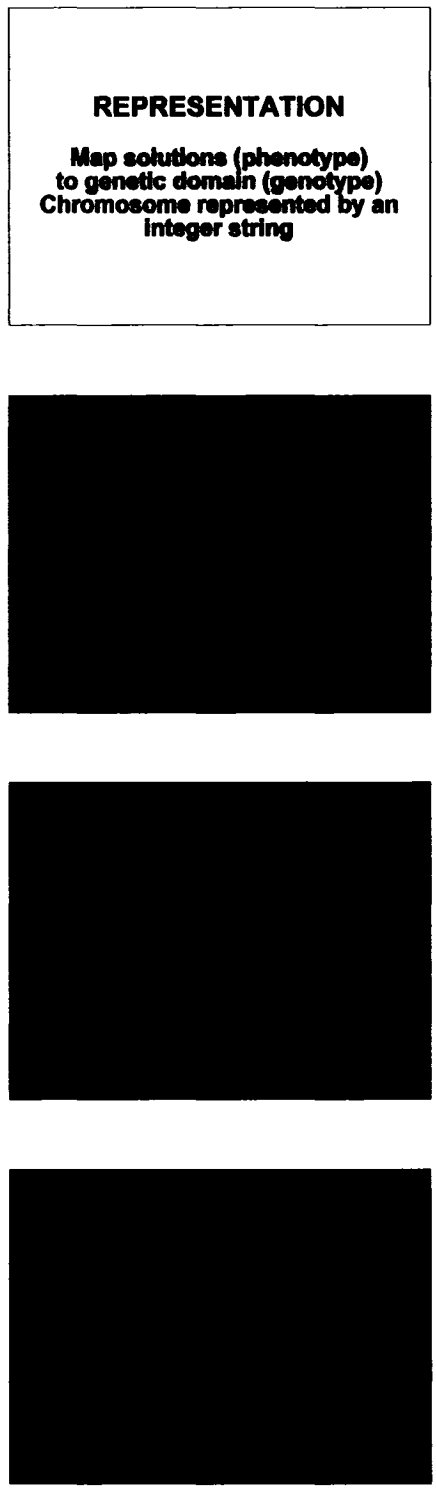

Figure 9: Overview of Genetic Algorithm components: A genetic representation that maps phenotype (solution domain) into genotype (genetic domain), a genetic algorithm process that evolves genotype based on natural selection plus applications of genetic operators such as crossover and mutation, a mapping process that maps genotype back to phenotype, and a fitness determination process that ranks the phenotypes for the purpose to be included in the next generation.

To make the description of the method clear, I will use as an example one of the queries used in this study (Case 1). In the example, the user is searching for an electronic form that is used by either the buyer or the seller for residential real estate in the province of Ontario. The goal is to find a web site containing an electronic form that is exactly the same as the hard copy form found in bookstores. The key words for this test case are: **"purchase"**, **"agreement"** and **"of"**. Table 4 shows how the GA terms used in describing each of these steps map onto Web search terms that describe the problem area. Both sets of terms will be used to discuss the four steps in more detail below.

Table 4: Mapping of GA terms to Web Search terms

Genetic Algorithm Terms	**Web Search Terms**
Genes	Key Words
Genotype	A specific query
Initial Genetic Population	Set of initial queries
Evolution	Generating new set of queries
Mapping genotype to phenotype	Submitting specific query to search engine
Phenotype	Output from search engine (list of URLs) to query
Phenotypic Population	Set of Outputs to all queries in population
Fitness	User ratings for each output

Representation

To use GA to optimize search engine performance, search engine operators and keywords are represented as genes, a specific chromosome or genotype is a specific combination of key words and operators, while the search results, the output web pages, are phenotype. There are several different ways to encode the chromosome. The most common way is to use binary encoding[21] which represents chromosome as a string one (1) or zero (0). In this study, I use a permutation encoding that assigns a unique number to identify a gene, or in this case a key word, such as "agreement". I call this form of representation an integer gene. Figure 10 shows how this form of representation works for the case of the homogeneous gene representation used in the unmodified (identified as "Current") GA representation. Each gene is associated with an integer number according to a permutation encoding scheme. Each gene represents a keyword. For example, the keywords <purchase, agreement, of> is associated with permutation number <0, 1, 2>.

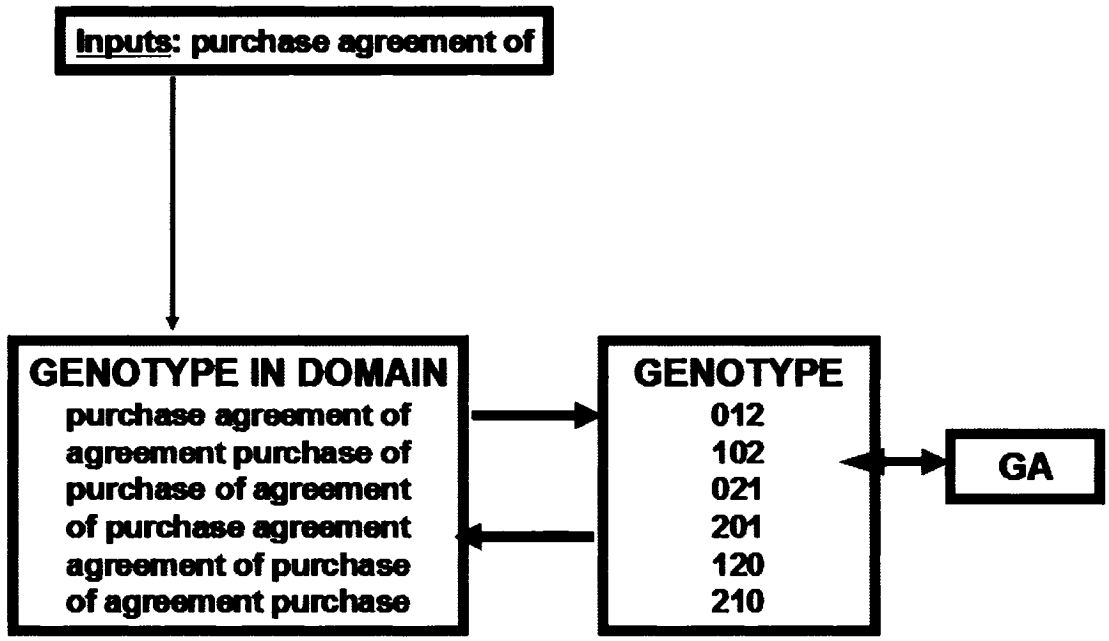

Figure 10: Current representation example of GA (called Structural in this study)

For modified GA, which includes both Regulatory and Deletion GA versions, shown in Figure 11, the representation is heterogeneous consisting of both structural and regulatory genes. Both types of genes are integer genes as described above, but have different roles, as seen in

Figure *12*. Regulatory genes, which are used to represent search operators, act to form relationships between structural genes, which represent key words. Using the same example described above, the structural genes represent the set of keywords <agreement, purchase, of> and the regulatory genes represent the set of search operators <[space], -, +> where operator [space] means OR, + means AND, and – means DELETION. In the right combination, these two types of gene form a genotype, or query, that will be effective as input to a search engine, producing a satisfactory search output, or a fit phenotype.

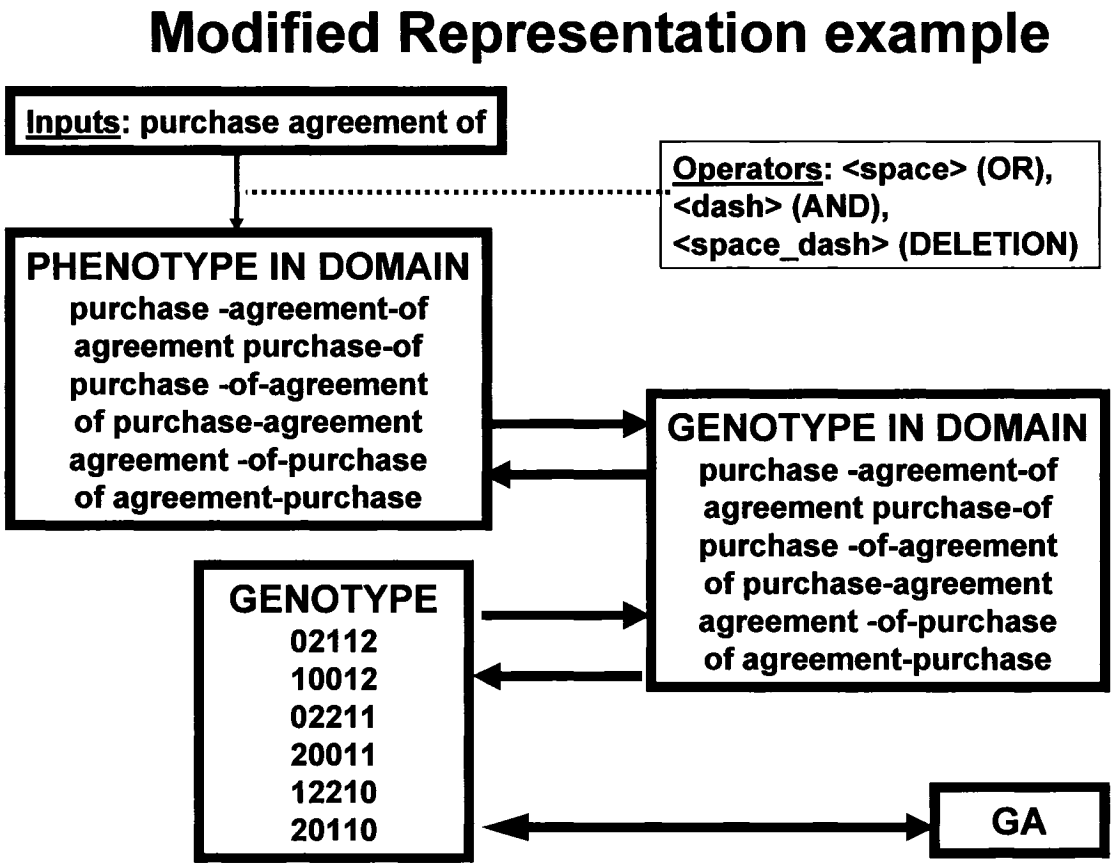

Figure 11: Modified representation example of GA (used in both Regulatory and Deletion GA <change from quotation to and>)

Note that the DELETION search operator is just another type of regulatory gene. As such it can operate on any of the keywords, removing them from the query. The deletion operator is not implemented as a regulatory gene in the first version of the Modified GA (Regulatory GA) but is added in the second version of the Modified GA (Deletion GA) to allow comparisons that will reveal the impact of this important component separately from other regulatory genes.

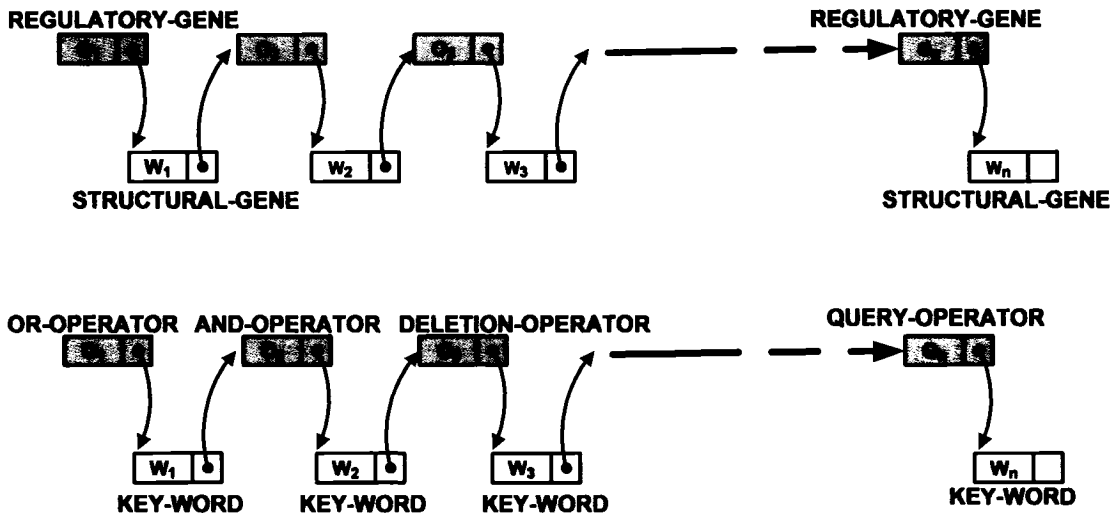

Figure 12: Interaction between structural and regulatory genes to create genotype in Modified GA representation

This comparison will be particularly interesting with respect to the test case (Case 2), based on key words <purchase, agreement, of>, but adding a silent gene, in this case the nonsense keyword "smith". The new test case now has a new structural gene set <purchase, agreement, of, smith> where the keyword *smith* makes no sense in relation to the goal of the user looking for an electronic purchase agreement. This represents the case where the user might enter an irrelevant keyword or one that might be used by the search engine in a different way than the user intended. Thus, Case 2 will look at the impact of the nonsense keyword in the presence of a deletion operator (Deletion GA), while Case 1 will provide information on the impact of the deletion operator (Deletion GA) in a situation where all keywords are relevant.

Genetic Algorithm

The Current or Modified representation acts as input to the GA, which is identical for both implementations, consisting mainly of the genetic operator crossover and mutation. I used a version of the basic GA algorithm published by John Holland in the form of software package named Java Genetic Algorithm Package (JGAP[b]).[22]. I selected this

[b] JGAP version is 1.0 RC1 and it has been used in the thesis since mid 2003

software package as the most flexible GA freeware available at the time with the entire software library implemented in Java. This flexibility allows the integration of other technologies from Apache and Google into JGAP, and allows a seamless flow of GA execution along with other program modules described below, such as Search Engine Interface, and Automatic Ranking (described in Testing GA). One of the most useful features of JGAP is an ability to create custom genes, genetic operators and fitness function.

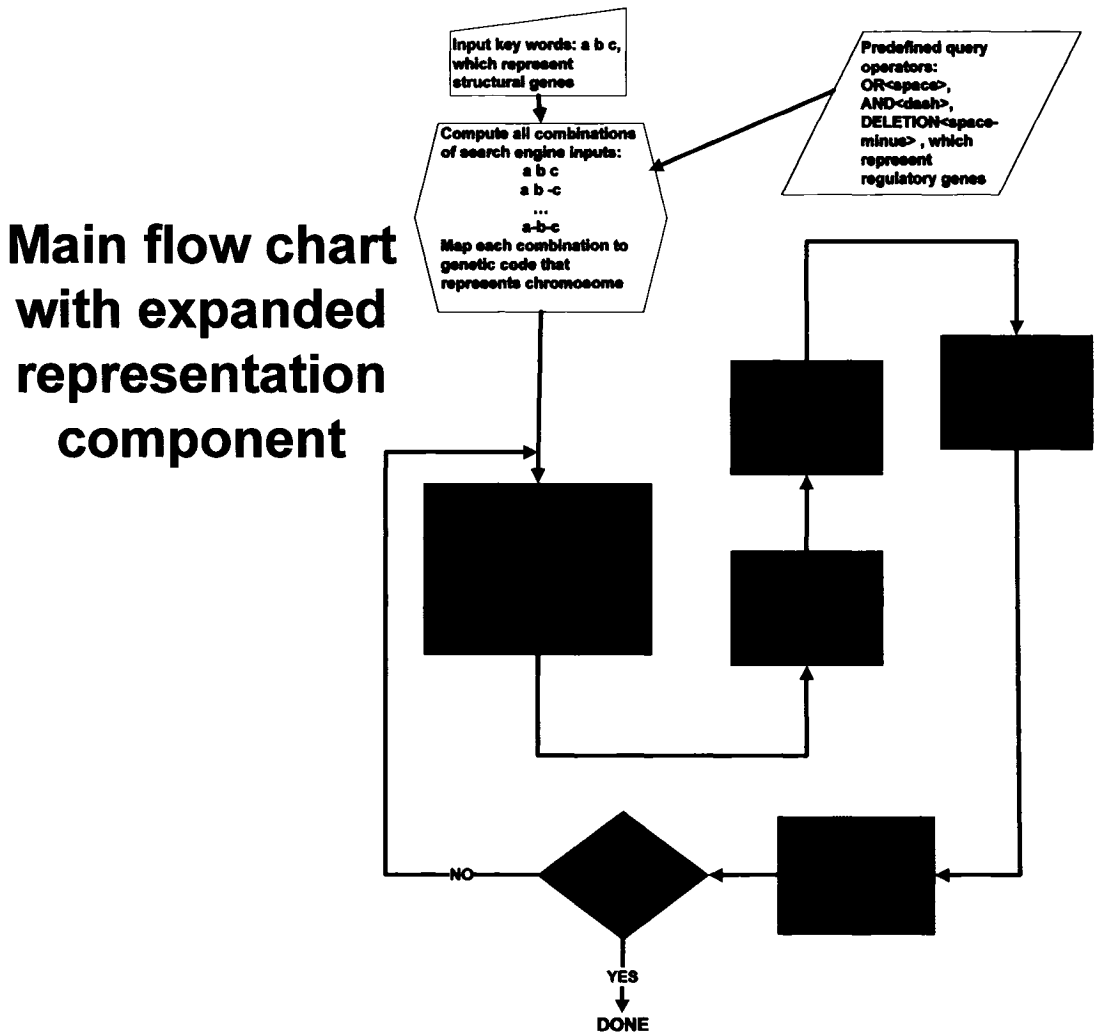

Figure 13: Implementation of GA

Figure 13 shows a high-level description implementation of a GA used in this study [although other kind Algorithms could be used here instead of GA but the scope of this thesis focuses on GA], which incorporates JGAP[c] into an overall process to produce the best queries for a given search engine. At a high level, block diagrams are used to represent a sub system or a sub process. Arrows indicates outputs from one block or sub system will be input to another block. The main core of the entire process is a loop that contains a number of sub processes that are executed repeatedly until either the input query is optimized or the number of generations (or evolutions) reaches the maximum limit. Starting the process is the representation of input query as genotype initial population. The loop starts with GA execution that applies GA operators on the current population. These genetic operators, crossover and mutation are applied to the chromosomes of the current population. The next population is selected from output of the GA execution using a ranking mechanism. In order to perform the ranking, the output genotypes from GA execution are mapped to search engine queries. The output of the search engine, the phenotype, will be evaluated by the fitness function to determine the fitness of each member of the population. The fitness values will be used in the weighted roulette selection algorithm to create the next population.

Crossover: A first sub-process applies genetic operator crossover and mutation on the current population. The first operator is single point crossover. It takes two (2) random chromosomes, picks a random locus[d] and swaps that gene and all genes to the right of that gene between the two chromosomes.

---

[c] JGAP information can be found at <http://jgap.sourceforge.net/doc/tutorial.html>
[d] Definition of locus: Gene location in the chromosome

Example of Crossover Operation:

Given two chromosomes #1:[purchase| -|agreement|-|of] and #2:[agreement| |purchase|-|of], the crossover operation starts as follow:

- Separate the regulatory genes from structural genes.
  Chromosome #1 contains regulatory genes [-|-] and structural genes [purchase|agreement|of].
  Chromosome #2 contains regulatory genes [|-] and structural genes [purchase|agreement|of].
- Then the chromosomes are encoded into integer string
  Chromosome #1 contains regulatory genes [2|1] and structural genes [0|1|2].
  Chromosome #2 contains regulatory genes [0|1] and structural genes [1|0|2].
- The genetic crossover is operated on regulatory genes separately.
  Regulatory genes:

  2|1 **0|1**
  —|— → —|—
  0|1 **2|1**

  Structural genes:

  0|12 **1|12**
  —|— → —|—
  1|02 **0|02**

- Now, chromosome #1 will be represented by integer string [1|0|1|1|2] and chromosome #2 will be represented by integer string [0|2|0|1|2].
- The combination of key word and search operator for chromosome #1 will be [agreement| |agreement|-|of] and for chromosome #2 will be [purchase| -|purchase|-|of].

Mutation: The second operator is mutation, which will alter one or more of the genes by replacing one integer gene with another. This operation is carried out separately for regulatory genes and structural genes.

Example of Mutation:

Given a chromosome of [purchase| -|agreement|-|of] will have regulatory genes [-|-] and structural genes [purchase|agreement|of].
The regulatory gene will represented by integer string [2|1] and structural genes will have an integer string of [0|1|2].
The mutation operator will alter regulatory genes to become [0|1] and structural genes will become of [2|1|0].
Hence, this chromosome will become [of| |agreement|-|purchase].

The mutation rate is calculated such that, on average, one gene will be mutated for every ten times a genotype is processed by this genetic operator.

Both crossover and mutation operators operate separately on regulatory and structural genes.

Mapping Genotype to Phenotype: The next sub process maps the genotype representation back to the phenotype. The first step in this sub-process is to map the integer genes back to the search engine input format, which is a combination of keywords and query operators. These are input to a specific search engine to produce output listings of URLs indicating web pages representing phenotypes. The content of each output page is fetched through its URL and ranked by a custom fitness function that will be described in detail below. The fitness value of a web page reflects the degree of relevancy to the user in achieving the objective. In the case of the example, the ranking relates to the value of the Web page output in locating an electronic version of the agreement of purchase form.

Selection: The final sub process in a loop is a selection of offspring to form the next population. The method to be used is weighted roulette wheel selection. Slots on the wheel are given to each chromosome based on its fitness value. During the selection, the wheel is spun and a chromosome is selected. This process repeats until all needed chromosomes are in the new population. Once the new population is formed, the cycle of GA execution is repeated.

To summarize, Table 5 shows a comparison of the implementations of the Current (Structural) and Modified (Regulatory, Deletion) GA within the programming structure used in this study.

Table 5: Summary of GA sub-processes for both the Current and Modified GA program modules

Current (Structural) GA	Modified (Regulatory & Deletion) GA
- *Encode GA representation **for input query keyword***	- *Encode GA representation **as a combination of input query keyword and search operators.***
- *Create an initial population*	- *Create an initial population*
- *Until the population become optimized*	- *Until the population become optimized*
o *Apply crossover operator **to genes***	o *Apply crossover operator **to regulatory and structural genes separately and combine the result.***
o *Apply mutation operator **to genes***	o *Apply mutation **operator to regulatory and structural genes separately and combine the result.***
o *Map Genotype to keyword-search-operator pattern*	o *Map Genotype to keyword-search-operator pattern*
o *Submit query and operator input to search engine*	o *Submit query and operator input to search engine*
o *Process output phenotype, which are web pages*	o *Process output phenotype, which are web pages*
- *Evaluate web page content using fitness function*	- *Evaluate web page content using fitness function*
- *Replace current population with new population*	- *Replace current population with new population*

Genetic Algorithm Parameters

Within the GA, there are a number of parameters that can be set or can be allowed to remain constant. To simplify the test executions and to ease the comparison between performance between an unmodified GA representation and the modified GA representations, all parameters are kept relatively constant except for number of simulation runs, the initial population and the number of generations.

The parameters that are not subject to experimental variation are mutation rate, the locus in the crossover operation and the method of selection. The mutation rate is set up as default value from JGAP software. It is a dynamic rate, which is calculated based on the size of the chromosomes population such that, on average, one gene will be mutated for

every ten times within the chromosome population[e]. Hence, the mutation rate is constant relative to the population. The choice of where one point crossover is located is randomly selected by computer execution. This parameter is treated the same way in all GA versions and test cases. The Weighted Roulette Wheel selection has no impact on the results on the results, as it functions in the same manner for both modified and unmodified GA.

The variable parameters in this study are discussed below and they are summarized in Table 6. These parameters are varied in the same way for both modified and unmodified GA.

Table 6: GA Parameters

Initial population	The initial population value set is <1, 5, 10, 15, 20>. This set of values is derived from trial and error runs as the best estimate values.
Generations	The number of generations (evolution) set is <1, 4, 8, 12>. This set of values is derived from trial and error runs as the best estimate values in a similar way as for "initial population".
Number of simulation runs	Value ranges from 10 to 1600. The range value is determined based on random trial number of experiments until the performance is stable or the standard deviation error is low.

Genetic Algorithm Output

GA outputs are the content of web pages, where the Web page URLs are produced by search engine in response to a given query input. The web pages are phenotypes and they are determined by the genotype (specific query) and environment (search engine).

Fitness

The content of web pages must be extracted and evaluated by a ranking sub process. A fitness value must be assigned to a web page based on how satisfied the user is with the content. The value ranges from zero (0) to ten (10) where zero means total dissatisfaction, ten means total satisfaction and any value between indicates partial satisfaction. Fitness value is assigned based on the presence of correct information

[e] Definition of class MutationOperator can be found at <http://jgap.sourceforge.net/javadoc/1.0/>

within the web site or the presence of information leading to the correct information. A small value is also assigned for a trusted domain such ".gov" or ".edu" as well as being in the right language.

Unlike other programmatic approaches, in this thesis, the fitness function is implemented as problem domain dependent. For each test case, rules on how to rank a document are used to determine the fitness value of the web page. This implementation cannot be generalized, as a single fitness function cannot be designed to handle all different types of queries or to model the satisfaction of all different users. However, it is generalized in that it can parse and evaluate any document within a particular test case, with some general rules as well as some that are specific to the user. Using non-generalized fitness function has minimal effect on the main questions of this thesis; that is, on the impact of GA modifications on GA performance, as the same fitness functions are employed in all test conditions. Thus I believe this approach is acceptable, within the scope of this thesis, to validate the main concepts of GA representation.

Rather than have users attempt to assign the fitness of all web page outputs generated within each GA execution loop, a fitness database was developed to assign a fitness value for a given pattern of regulatory and structural genes. To do this, a fitness value is determined from the contents of returned output pages for every possible combination for each test case. This process was automated using a set of rules on how to determine relevancy based on user input. Simple rules involve the occurrence of certain keywords, phrases or images in the content. More complicated rules involved determining if the web page provides appropriate information to direct the user on how to obtain more information. These rules were implemented in Java code so that the fitness values can be automatically assigned to a large number of web pages output from submissions of all possible combinations of keywords for a particular test case. These values are then linked to the keyword combinations themselves. These combinations are output with each evolutionary cycle as a population of genotypes from the GA. Figure 14 provides a graphic summary of this program.

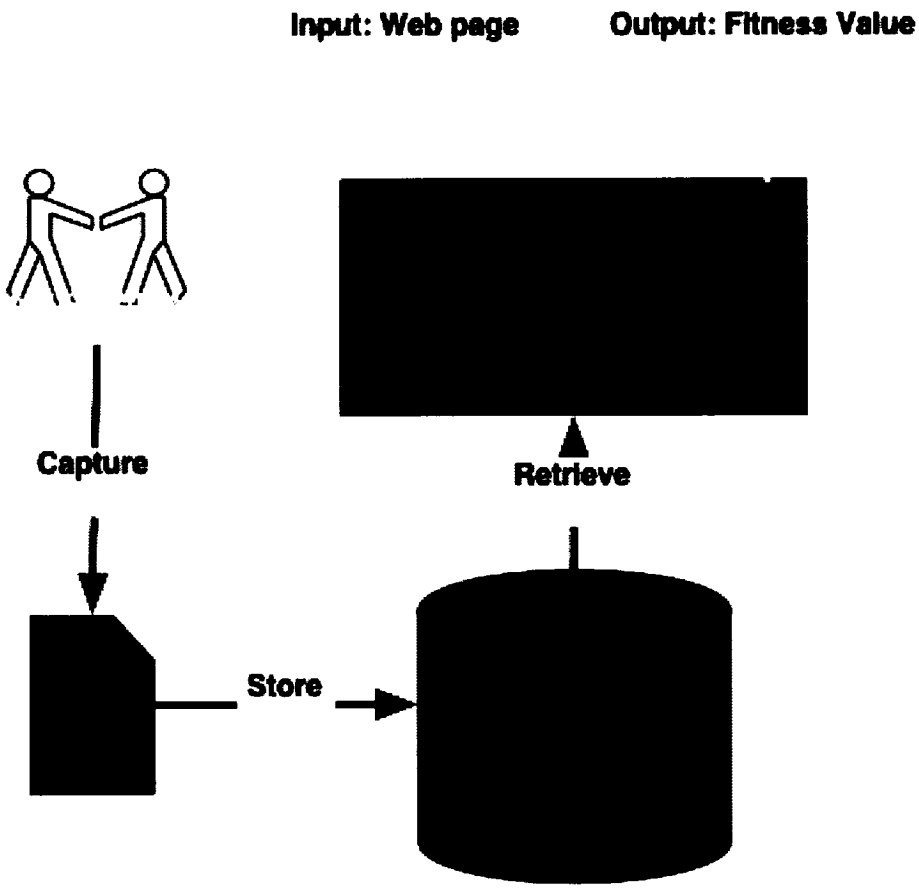

Figure 14: Ranking mechanism

CHAPTER 3: TESTING GA

Figure 15 provides a flowchart of how the components of the GA described above work together to optimize search queries. First, all input keywords are mapped to structural genes. For the modified GA representations, the regulatory genes consist of three search engine operators AND, OR and Deletion. All combinations of structural and regulatory genes are computed and are mapped to form all possible genotypes. An initial population is drawn and the GA execution is repeated until either a certain number of generations or an optimized query is reached. Within the GA execution process, parents are selected based on fitness using a weighted roulette model. The genetic operator crossover and mutation are applied to produce new offspring. Note that in the case of the modified GAs, crossover and mutation are applied on structural and regulatory genes separately and the output results are combined to produce the genotype. The genotypes of the new offspring are translated to search engine queries that are submitted to a specific search engine resulting in a list of output pages. The content of these pages is then extracted on the Web and are fed to an automatic ranking system where a fitness value for each set of output pages is determined automatically using rules supplied by the user. The fitness values associated with each output/query are also stored in the fitness database.

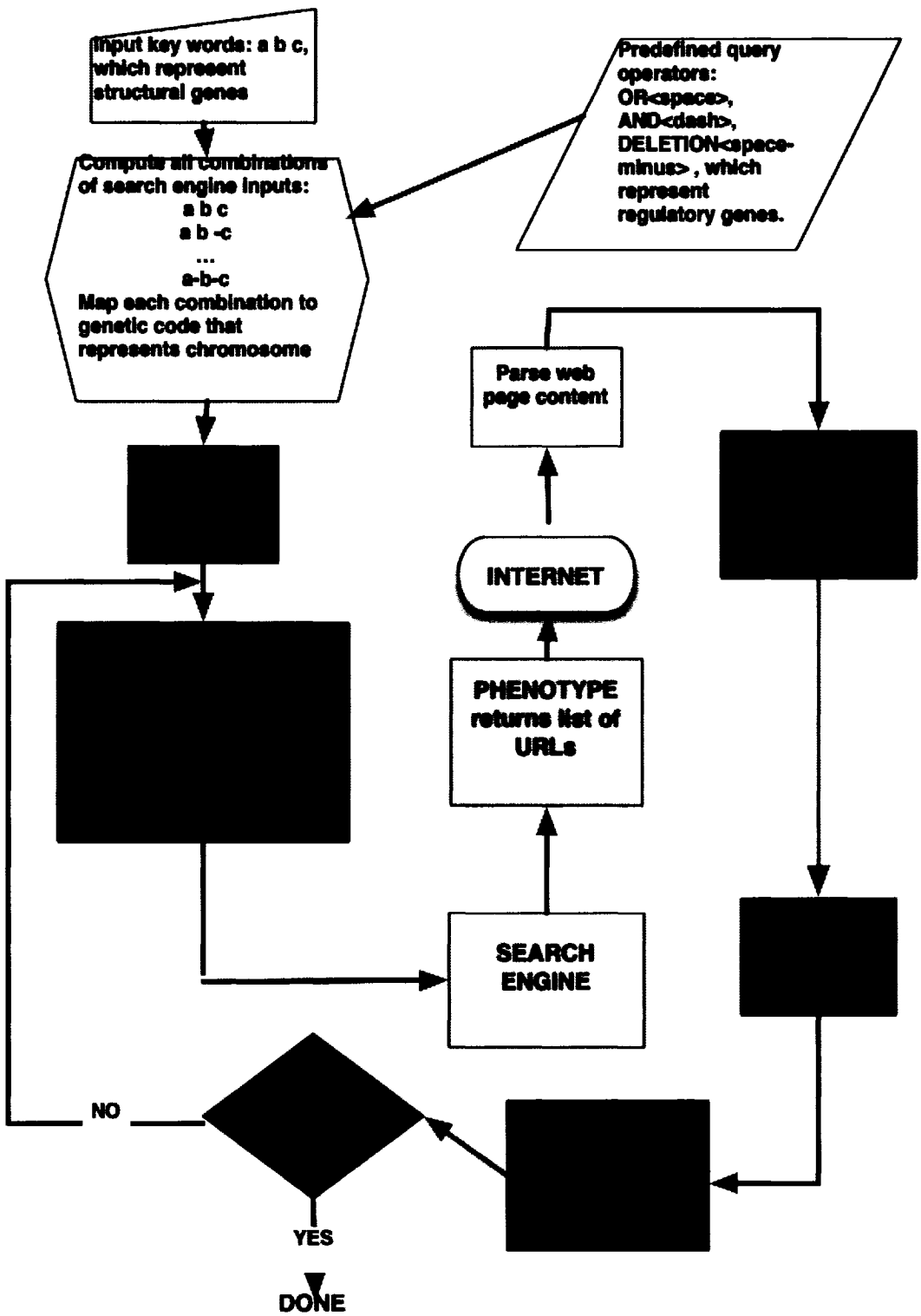

Figure 15: GA real-time implementation in detail

Real Time Implementation

The best possible way to compare the performance of the Structural (Current) Regulatory (Modified) and Deletion (Modified) GA in optimizing Web search is to analyze the output of their execution online and in real time. This process is described in Figure 16. The overall architecture consists of three main subsystems. The GA subsystem is responsible for executing the unmodified GA and modified GA to optimize the search engine query. Inputs to the GA subsystem are keywords and parameters such as initial population and number of generations (evolutions). Keywords generated by specific users are entered into GA subsystem. The Search Engine subsystem consists of one of the search engines; in this study, Google, Yahoo or Netscape and the corresponding interface. The Search Engine subsystem takes input keywords and search operators from the GA subsystem and produces a list of URLs, which represent intermediate outputs. The list of URLs is then input into an Automatic Ranking subsystem. The automatic ranking system produces fitness values, which will be used in the GA subsystem to determine the next generation. User-defined rules are input to the automatic ranking system and used to rank the output Web pages derived from the URLs.

In the overall architecture of the real-time implementation; only the Search Engine subsystem and Automatic Ranking subsystem would be real-time and online.

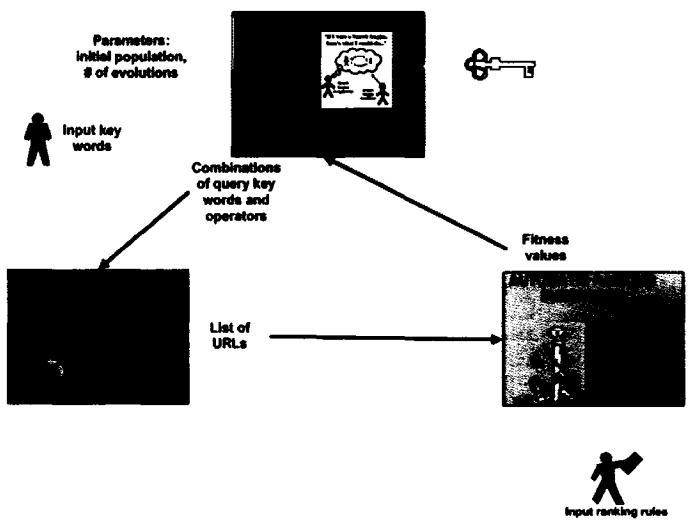

Figure 16: Real time implementation of GA: Overall Architecture

Table 7 describes in detail how those two subsystems fit into the algorithm. The highlighted section of Table 7 shows the tasks that would be real-time and online. The interfacing to the search engine and inquiring about user satisfaction for evaluating the output web page are tightly integrated to GA execution in the real-time mode.

Table 7: Real time implementation of test execution

Integer string	←	*Encode genetic code – keywords and search operators*
		Randomly select an initial population
Loop: While generated queries are not optimized		
Integer string	←	*Crossover – GA execution*
Integer string	←	*Mutation – GA execution*
Search engine input	←	***Mapping – Genotype***
Search engine	←	***Interfacing – Search engine input***
URLs	←	***Output response – Search engine***
Output web pages	←	***Crawl The Web – URLs***
Fitness values	←	***Ask user satisfaction – phenotype***
New population	←	*Selection – Fitness values*
End Loop		

However, for several reasons, an on-line implementation is not practical. First of all, the large size of the Web makes it very time consuming to submit all the queries generated during the GA process in real time and receive the output back. Secondly, the fact that this process takes time means that the Web can change during the GA and between GA runs. The dynamic nature of the Web in this case works against a valid comparison to the two GA approaches. Thirdly, most search engines are protected against intruders or the appearance of intruders. The type of input that would be generated by these programs is very likely to cause such a reaction. Therefore, the execution was implemented using simulation technique.

Implementation by Simulation:

To avoid these problems, the real time components of the programs were executed separately from August 2003 to September 2004 and the output provided to the GA in intermediate output form so that the GA components can be run in a simulation mode. The programs that were executed separately were the Search Engine subsystem and the Automatic Ranking subsystem. In simulation execution mode, the intermediate outputs

from these two systems, consisting of a list of URLs and an associated fitness value generated by every possible query combination, are used in GA execution to simulate the output that would be generated from submitting on-line queries to Web search engines in real time. This simulation of the interactions with the Web is more efficient, overcomes problems with search engines and captures Web information at one point in time that can be used with all different test conditions.

Figure 17 shows the GA simulation implementation in detail. The difference between Figure 15 [real-time implementation] and Figure 17 [simulation implementation] is in the use of fitness database as an implementation of fitness function. The comparison between Table 7 [real-time implementation] and Table 8 [simulation implementation] is similar.

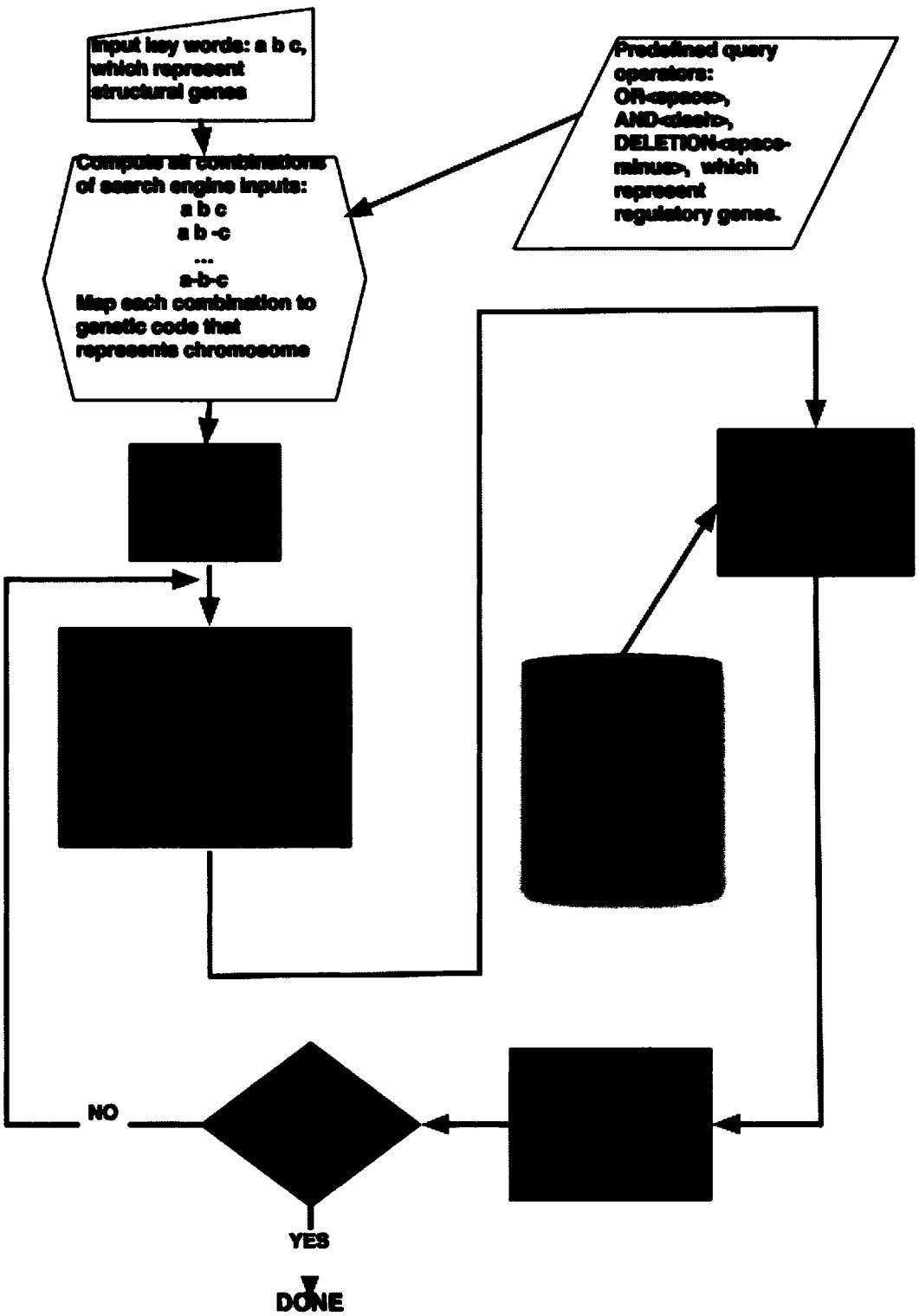

Figure 17: GA simulation implementation in detail

The GA sub-system now can be executed in the simulation just as it would be in real time. Comparing Table 7 and Table 8 below demonstrates that the main difference between a test execution in real time and a test execution in simulation is the way that the fitness values are obtained. In the simulation, fitness values are retrieved from an offline fitness database, avoiding the lengthy process to capture output web pages to perform an online ranking in real time. In the simulation, the only real-time component, as highlighted in Table 8, is the collection of information about fitness values for the fitness database.

Table 8: Non real time implementation of test execution in simulation mode

Integer string	←	*Encode genetic code – keywords and search operators*
		Randomly select an initial population
Loop: While generated queries are not optimized		
Integer string	←	*Crossover – GA execution*
Integer string	←	*Mutation – GA execution*
Fitness values	←	***Fitness database – phenotype***
New population	←	*Selection – Fitness values*
End Loop		

Implementing the Fitness Database

A fitness database is built to provide GA subsystem a capability to retrieve the fitness value when the fitness function is executed. This database is implemented as a table in an Oracle database. During the simulation execution, fitness data could be retrieved directly from database itself via JDBC protocol. Another faster way is to load all fitness data from a database export file in the beginning of the execution. The combination of keywords/search-operators and fitness value are stored in the fitness database as shown in Table 9.

Table 9: Example of how fitness values are stored in database for <purchase, agreement, of>

purchase agreement of	0	agreement of purchase	0
purchase agreement-of	0	agreement of-purchase	0
purchase agreement -of	0	agreement of -purchase	0
purchase-agreement of	0	agreement-of purchase	10
purchase-agreement-of	0	agreement-of-purchase	7
purchase-agreement -of	0	agreement-of -purchase	0
purchase -agreement of	3	agreement -of purchase	0
purchase -agreement-of	3	agreement -of-purchase	0
purchase -agreement -of	0	agreement -of -purchase	3

Implementing the Search Engine Subsystem

The Search Engine Subsystem is based on a separate program that automatically interfaces to search engines to get a list of URLs. Then it crawls the Web to obtain the web page content for each URL. This program uses Apache Web technologies to interface to search engine and to crawl the Web using simple HTTP protocol. The difficulties it must face are listed below in Table 10, along with the solutions I developed to resolve the issues.

Table 10: Outline of issues relating to the development of a programmatic interface to web sites along with the solutions adopting in the coding

Issues	**Solutions**
High resource and time requirements to complete Web tasks.	Create multiple instances of the program that can be executed in parallel, sharing the workload.
The content of the web pages are dynamic, but I need a "snapshot" of web information.	A large number of instances of the program are launched at the same time.
Connections to web sites are often lost or exceptions occur.	Program is designed to self-recover and is able to continue from where it left off.
Hostile reaction from search engine thinking it is facing an intruder, hacker, spammer or worm.	Design sophisticated delay so the program imitates a user entering queries on a keyboard.
Exceptions happen during Web crawling and interfacing with web sites.	Improve the programs so they are autonomous and able to restart themselves in the case of an error or web site program crash

The issues with search engine interface are the size of the Web, the security aspect of a web site, and the time and resources needed to complete the tasks. To resolve these issues, I executed multiple instances of the search engine interface program in parallel. The workload required to complete the tasks is partitioned among those instances. These instances must be autonomous as much as possible. If they die, they must try to re-launch themselves. I also increased the robustness of these programs so that they can recover from different types of errors that commonly occur while searching the Web, making them autonomous. In case of failure, the instances will continue the tasks given to them from where they stopped the last time. The last issue is the protective reactions

of search engines and web site administrators against hackers, intruders and outside attacks such as worms or denied-of-service (an attack that overwhelms the web site network with requests to prevent the web site from performing normal functions). Therefore, it is common for any web site, including search engines, not to trust programmatic access. As described in Table 10, I resolved this issue by making my program execution look like a person typing query input on computer keyboard. I programmed the execution instances to slow down after submitting request. However, this approach makes the execution become longer than normally required. So I scheduled this program to have as much instances as possible and to execute on multiples servers from different locations (to simulate queries coming from sources).

Table 11 describes how the Search Engine subsystem works. Given a set of keywords, all possible combinations of keyword and search operators are computed as input to a specific search engine. The output URL list from that search engine in response to a specific combination is used to allow my program to craw the Web and extract the web page content. The web page content for each URL in the list is then input into the Automated Ranking System. This process continues until the URL lists for each combination have been obtained. The process is repeated for each search engine.

Table 11: Outline of process by which Search Engine subsystem collects output web pages

Search engine input	←	*Compute all patterns – keywords and search operators*
Loop: For each pattern of keyword and search operator		
Search engine	←	*Interfacing – Search engine input*
URLs	←	*Output response – Search engine*
Output web pages	←	*Crawl The Web – URLs*
End Loop		

Implementing the Automatic Ranking Subsystem

The Automatic Ranking subsystem assigns a fitness value to each Web page associated with a URL in the search engine output listing and then calculates an overall fitness value for each URL listing returned for a given input query. This value is returned to the GA subsystem.

This method of evaluating fitness is not standard. Precision and Recall are standard measurements to evaluate information retrieval systems. They normally involve the comparison of a retrieved document to the entire set of documents containing the keywords. Thus, the implementation of this method for optimizing web search would require the processing of very large set of documents available on the web[23]. Therefore my implementation focuses on fitness values based user satisfaction. While this approach is practical, this could be a limitation for a real time system with a large number of users.

To obtain the ranking rules for each test case, I interviewed the user who supplied the test case keywords to see how he or she would rank a web page regarding the relevance of it's content. The assigned fitness value is based on this relevance as well as how easily the information can be found (how many links required) and how trustworthy the web site is (i.e. .gov or .edu are considered more trustworthy than .com or .org). The ranking rules are implemented in Java code, which is used to evaluate to represent the ranking rules. This Java code will be used by to evaluate web page. The average fitness value for the top five URL listings in each search results are used to compute a fitness value that is associated with the search results of a particular combination of keywords and operators. An example is given below in Figure 18.

```
if (
   (lcPage.indexOf(
    new String("AGREEMENT OF PURCHASE AND SALE FORM").toLowerCase()) !=
    -1)  && (
   (lcPage.indexOf("orea") != -1)          ||
   (lcPage.indexOf(
    new String("Ontario Real Estate Association").toLowerCase()) != -
    1))
     )
    result = 10;

if (lcPage.indexOf("agreement of purchase and Sale") != -1)
{
    temp = 2;
    if (result < temp)
         result = temp;
}
```

Figure 18: A sample of Java code translated from ranking rule

Test Cases

Three test cases were used in this study. In the first test case (Case 1), which is described in the previous section, the goal is to search for an electronic form that is used by either the buyer or the seller for resident estate in the province of Ontario. The key words for this test case are **purchase**, **agreement** and **of**.

The second test case (Case 2) is aimed to test the idea of the impact of a silent gene. The keywords are the same as those used in the first experiment but with an addition of nonsense keyword "smith".

This third test case (Case 3) is a simple search for a hotel in Montreal. The hotel should be located in walking distance from Place Des Arts in Montreal and the price should under $ 130.00 Canadian. The suggested key words from the subject are hotel, Place, Des, Arts. This test case is special. Place Des Arts is a very well known location in the downtown of Montreal. Located within walking distance to both Chinatown and to the heart of Montreal, it is an ideal location for both tourists and business travelers. The subway station is right under the Place Des Arts, and one can quickly go from there to anywhere in the city. The popularity of Place des Arts makes it a very good example for

a search of a place or an item where there is immense Internet advertising about it (see Figure 19).

Each of these test cases was used in the execution of the three GA conditions: Structural, Regulatory and Deletion for three search engines: Google, Yahoo and Netscape.

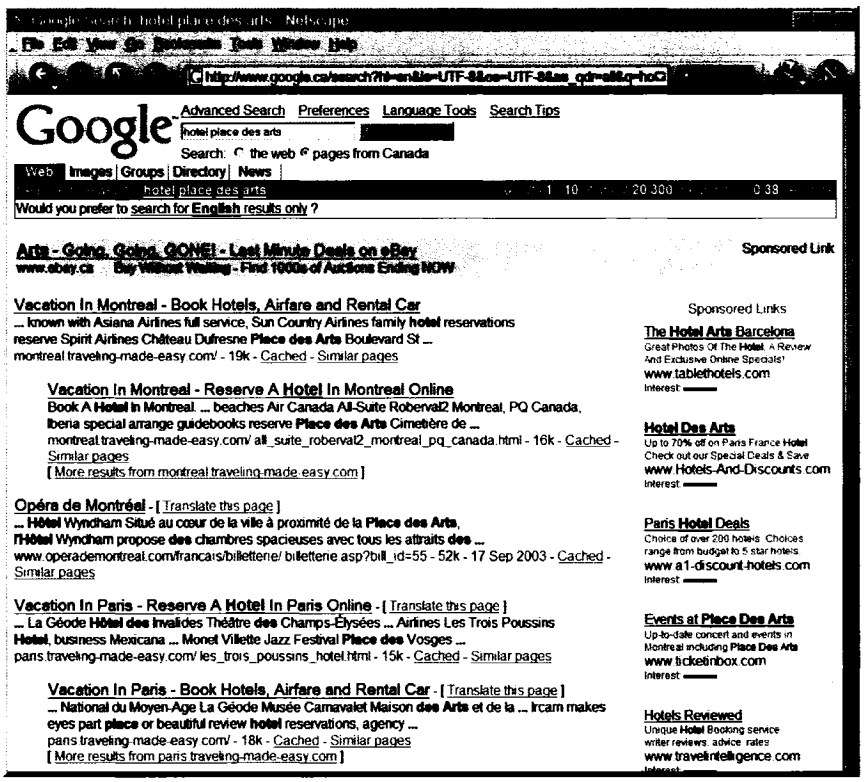

Figure 19: Third experiment using keywords: Hotel, Place, Des, Arts.

CHAPTER 4: RESULTS

The results of this study are reported below in three sections. The first section deals with the performance of the components of the simulation; the second section describes the impact of the simulation parameters on the stability of the GA results and the final section compares the performance of the (Structural) to the performance of the two modified versions of the GA (Regulatory and Deletion) in optimizing search queries.

Performance of Simulation Sub-systems

As described in the methods section, in the course of implementing programs for test executions, several technical issues had to be overcome with respect to interfacing with the search engines to collect data in response to queries and automating the user ranking process. Without special modifications, The Google search engine shuts down all connections originating from my IP address after three to four minutes of test executions. Of the three search engines, Google is the most sensitive one in terms of detecting intruders.

By developing methods to simulate user behavior, the search engine interface was able to remain continuously connected and successfully complete the database URL listings for all possible queries generated by the three GA conditions for all three test cases. These modifications were successful in preventing being shut out as an intruder and allowed the program to maintain a continuous connection and to download information for as long as 48 to 72 hours, depending on the simulation.

If a single instance had been used to execute the interface program, the entire program would have taken 3 or 4 days to complete. During this time, it is quite likely that the contents of some of the web pages would have changed, making it difficult to capture a "snapshot" of the information that will be returned to all possible queries. By using a distributed computation approach to partition the tasks to focus on single sets of key words and operators and sharing the workload between multiple instances on multiple computers, I was able to reduce the average time spent on search engine interface to less

than 24 hours to 36 hours. The only exception is when there are events on the net, such as sites being closed for maintenance, that slow down the sequential process of assessing the results of each query.

Similarly, in my initial attempts, web-site crashes or time-outs also slowed the completion of collecting the information on the URLs and web pages. This was particularly a problem when using a distributed computation approach to divide up the interface tasks. On average, the interface program was halted due to web-site problems at least two or three times per run in the initial program. By introducing self-recovery components, the multitasking system was able to recover from almost all of the web site problems and, in its final form, successfully completed all executions (all possible query combinations for the three GA conditions for the three test cases and three search engines) without crashing. Self-recovery components are particularly important for programs such as this one that require the completion of a large number of sequential tasks.

An automated ranking system was developed to undertake the time-consuming task of ranking the top five web pages returned for each of all possible combinations of search queries. By comparing the results of this ranking process with the results of users manual rankings for a sub-set of search results, I determined that the agreement was good, ranging from 70 percent to 100 percent, depending on the test case.

Performance of GA Simulation

The performance of the Current and Modified GA was evaluated using three test cases ("purchase, agreement, of"(Case 1); "purchase, agreement, of, smith" (Case 2) and "hotel, Place, des, Arts" (Case 3) and three search engines (Google, Yahoo and Netscape). The Current GA includes only structural genes and is referred to in the results and discussion as the Structural GA condition. The first version of the Modified GA uses representations of both structural and regulatory genes and is referred to as the Regulatory GA condition. The second version of the Modified GA includes a deletion operator in the regulatory genes and is referred to as the Deletion GA condition.

An example of the simulation results are shown in Figure 20. On the left side of the figure, the fitness values for each of 10 simulations for the Regulatory GA using Case 1 (purchase agreement of) with Google are shown as the GA evolves from the first to the twelfth generation. In this case, the initial population was 20. This graph shows how the performance the GA varies within a simulation and from one simulation run to another. Note that there is considerable variability between initial simulations, although, on average, there is a steady increase of fitness over generations. These results are summarized in the right panel using the average fitness values across all 10 simulations for each generation. The standard deviation is given for each average value as an indication of the variability of the individual values around the average. Note that the standard deviation is greater for the initial generations. Average fitness values and estimates of variability will be used in all subsequent graphs to describe the results of this study.

The results for this study are reported in two parts. The first section describes the results of evaluating different GA parameters to determine the best values to use for the comparison of the two GA approaches. The second section describes the results of the comparison of the two GA approaches for the three test cases for three different search engines.

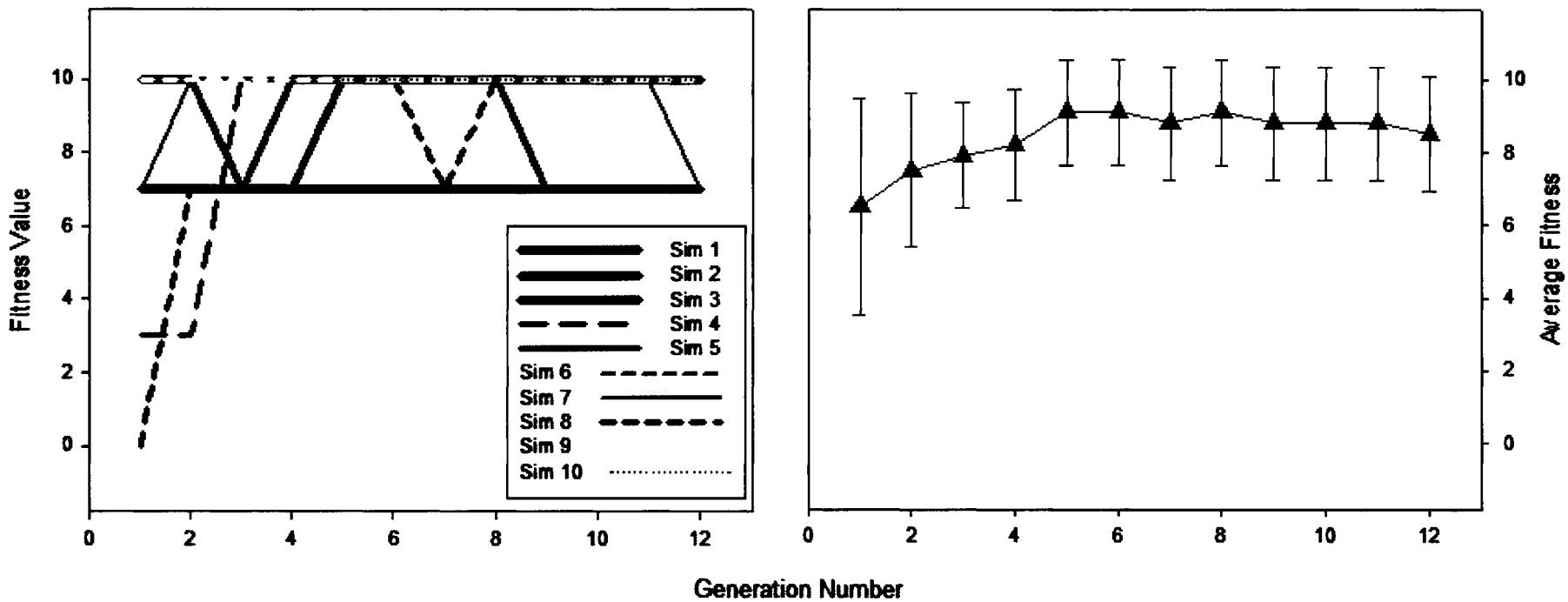

Figure 20: Example of results for individual simulations of the Regulatory GA condition (left panel) showing 10 simulations of Google-Case 1. A summary of these results is presented (right panel) with average fitness values and standard deviations for each generation.

Table 12: Results of GA parameters choices

Parameters	Values
Initial population	If the size of the initial population is too small, the number of choices for evolution is limited and there are too few possibilities to perform crossover. However if the initial population is made too large, the GA execution may require more time and resources to complete the execution.
Number of generations	The parameter has a similar impact on performance as the "initial population" parameter such that the performance is better with larger number of generations but there is a computing cost with it.
Number of simulation runs	With too few simulations, the average performance of the GA system will not be stable. Increasing the number of simulations will improve performance, but there is a computing cost associated with the increase.

Impact of Simulation Parameters

GA execution has multiple variable parameters: initial population, number of generations and the number of simulation runs. Table 12 shows how values of GA parameters are likely to affect the test results. To determine the best values for these parameters, the number of simulation runs, the size of the initial population and number of generations were varied independently while measuring the average fitness value and the variability.

The Effect of the Number of Simulations: Figure 20 shows that there is considerable variability in the results of 10 simulations. Increasing the number of simulation runs decreases the impact of this variability and increases the stability of the results, as shown in Figure 21. This figure shows the results for the three GA conditions for simulation numbers from 10 to 1600. The results shown in these graphs are for the 12 generations with an initial population of 20. The results for each case (3) and search engine (3) are shown in a separate graph (9).. The standard deviation is given for each average value as an indication of the variability of the individual values around the average. While the results are generally more stable for simulation runs of 400 or greater, there is still some improvement in some cases up to 1600. Thus, for the purposes of comparing the results of the three forms of GA, simulation runs of 1600 are used.

It is worth noting that, in most test cases, the performances for the Regulatory and Deletion GA are similar and better than performance for the Structural GA. Overall performance varies with the case, as does the variability in performance. In most conditions, the Structure GA condition produces 0 fitness levels with 0 variability, indicating that the fitness value for queries based on homogeneous representation do not evolve and remain at 0 for all simulations in most conditions. The exception to this is for Case 3 with Netscape. In this condition, the Structure GA produces higher average fitness values although the variability for these results is also very high even for 1600 simulation runs.

In terms of GA performance and variability across simulations, Google-Case 1 and Netscape-Case 3 show considerable differences. Therefore in the discussion of the impact of the other parameters on GA results, these two cases will be used as examples.

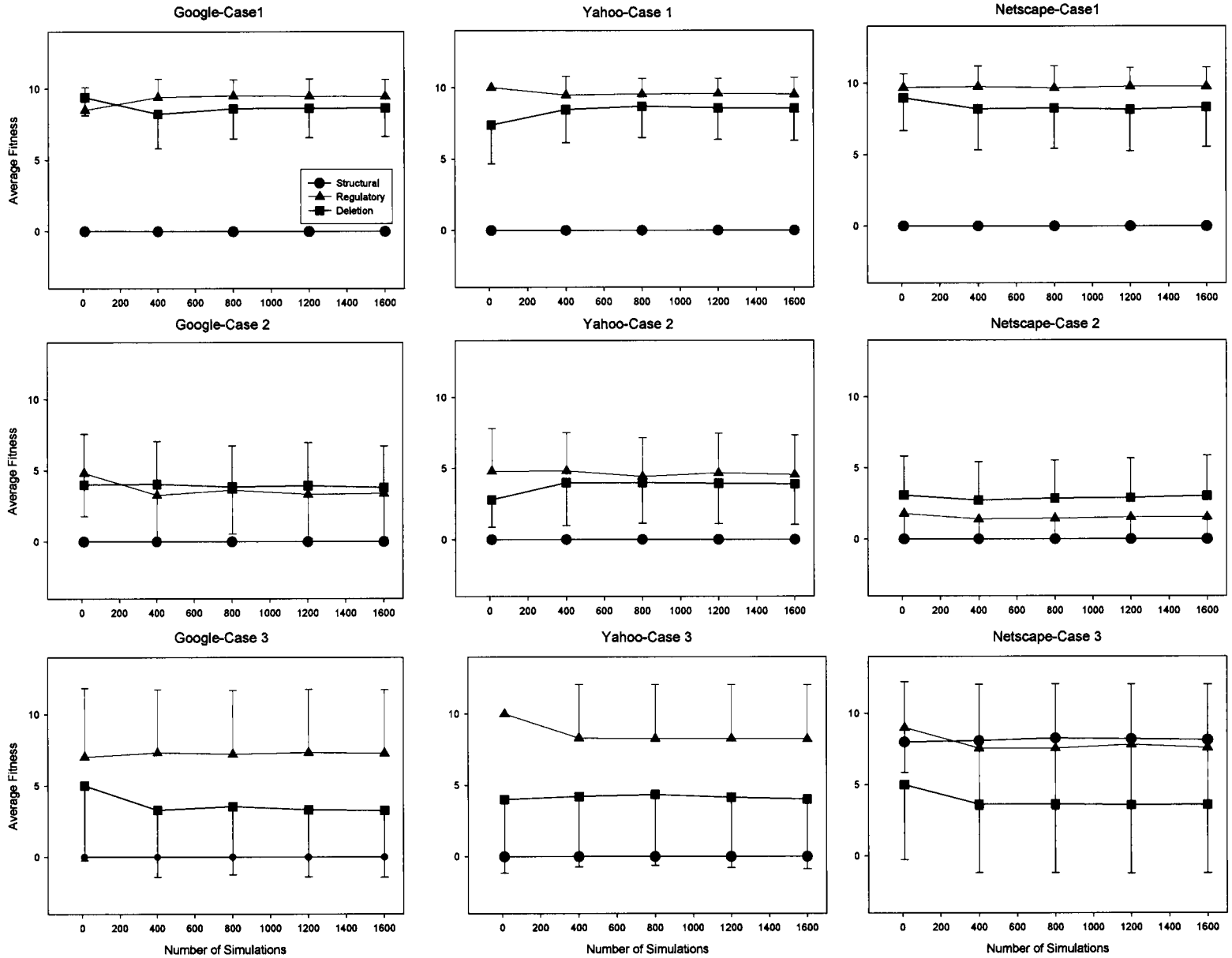

Figure 21: The effect of number of simulations for all cases (columns) and search engines (rows). Results for all three GA conditions shown in each graph.

Effect of Number of Generations: Figure 22 shows how the performance of the GA varies with the number of generations. . The standard deviation is given for each average value as an indication of the variability of the individual values around the average. The left panel of Figure 22 shows the results for the Structural, Regulatory and Deletion GA conditions for Case 1 with Google. The right panel shows the same thing for Case 3 ("hotel place des arts") with Netscape. In both cases there were 1600 simulations using an initial population of 20. In both graphs, average fitness increases with the number of generations and the standard deviation decreases. The fitness appears to level off between 10 and 12 generations, although not completely. This indicates that the results become more stable for these generation numbers, although it should be noted that 12 generations was not sufficient to improve the fitness values that resulted from the Structural GA in Case 1 with Google. However, in the only conditions where the Structural GA produced non-zero results, Case 3 with Netscape, the average fitness value does increase as the number of generations increases. For Case 3 with Netscape, variability remains high, indicating that, even with 12 generations, the fitness value of individual simulations varied widely in these conditions. Based on these results, 12 generations were used to derive the most stable performance data for all conditions for the purposes of comparing the results of the three forms of GA.

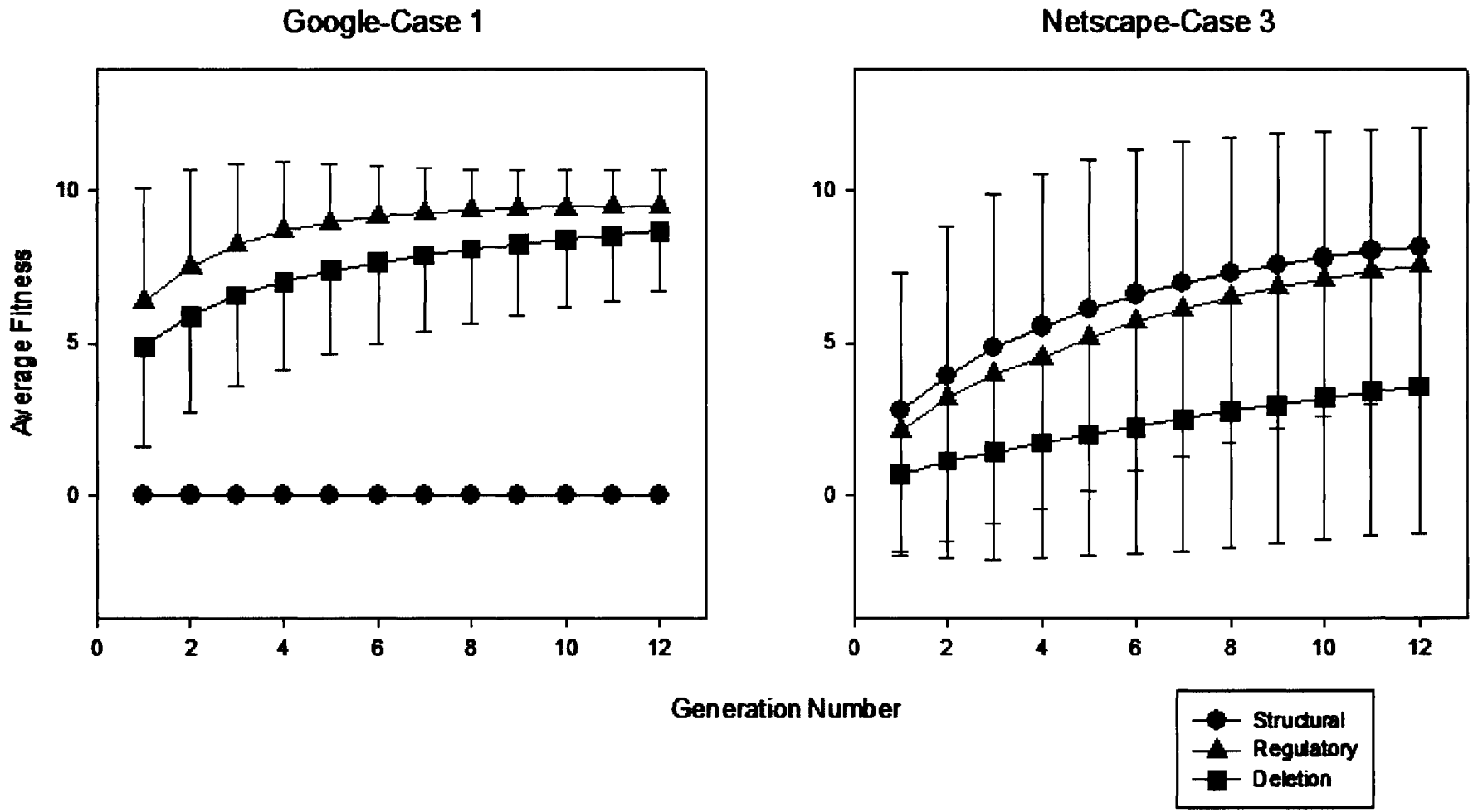

Figure 22: The effect of number of generations for 2 examples: Case 1 with Google (left panel) and Case 3 with Netscape (right panel). The other parameters are initial population=20, simulation runs=1600. The three GA conditions are plotted in each graph, showing the average fitness and standard deviation.

Effect of Initial Population Size: Figure 23 shows the results for the three GA conditions for two cases and two search engines across the four different values for initial population size. . The standard deviation is given for each average value as an indication of the variability of the individual values around the average. The left panel shows results for the test Case 1 with keywords "purchase, agreement, of" using the Google search engine. As in the other graphs, Y-axis shows average fitness value range from zero to 10. The number of simulation runs is kept constant at 1600 and the number of generation is set at 12. The X-axis shows the size of initial population, which ranges from 5 to 20. In Case 1, for both Regulatory and Deletion GA, performance improves as the initial population size increases, while the results for Structural GA remain unchanged at a fitness value of zero. The right panel shows the results for Case 3, "hotel, place, des, arts" with Netscape. In this situation, the performance of all three of GA improves with initial population size. In both panels, the variability also decreases for most of the GA conditions, although, as with previous graphs, variability is higher for Case 3 with Netscape. Based on these results, an initial population size of 20 was selected for the comparison of GA performance.

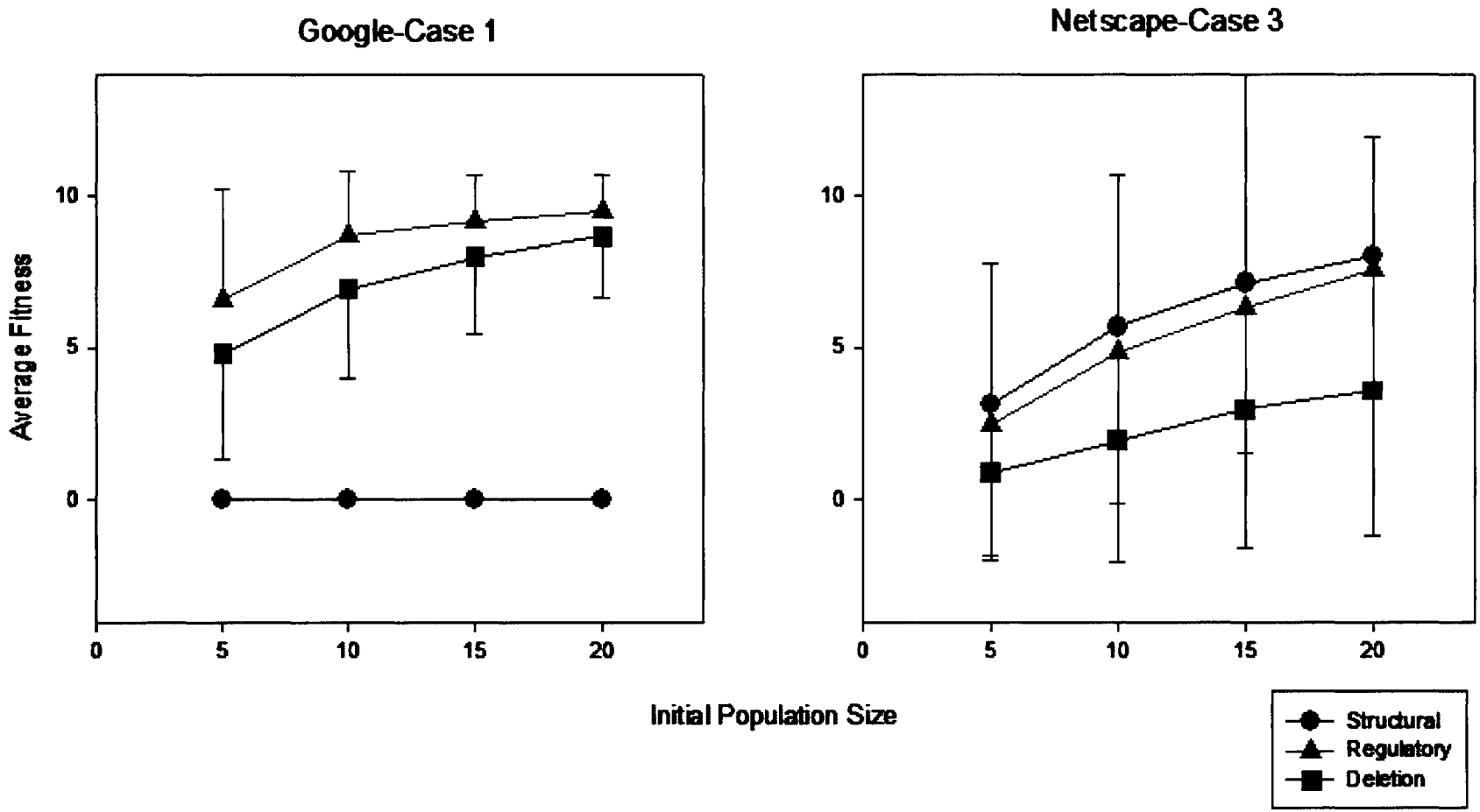

Figure 23: The effect of initial population size for 2 examples: Case 1 with Google (left panel) and Case 3 with Netscape (right panel). The other parameters are generations=20, simulation runs=1600. The three GA conditions are plotted in each graph, showing the average fitness and standard deviation

Comparison of GA Performance

Figure 24 shows a comparison of the results for all conditions run with the optimal simulation parameters: 1600 simulation runs, 12 generations and a population size of 20. Each panel shows the results for one case (rows) and one search engine (columns). The results for all three GA (Structure, Regulatory, Deletion) are shown in each panel. For each GA, the average fitness value is graphed along with the standard error, which provides an estimate of the variability of the average fitness values if I was to repeat this experiment. In most cases the standard error is very small and barely visible on the graph. The size of the error indicates that the results are very stable and repeatable; thus any differences between the performance values plotted are likely to be real and significant.

Table 13 shows a summary of the statistical analyses of the results shown graphically in Figure 24. The complete statistical analyses are presented in Appendix A. Due to the fact that the data were not normally distributed, a Kruskal-Wallis one way analysis of variance on ranks was performed on the result for each combination of test case and search engine. In every case, the differences between the Homogeneous Representation of Genes and Heterogeneous Representation of Genes with or without Deletion are statistically significant [$P<0.05$]. Further analyses were not possible due to the fact that data were not normally distributed.

Table 13 Statistical Analysis of Results using 1600 Simulations with Generations =12 and Initial Population Size = 20

Condition	*Kruskal-Wallis One Way Analysis of Variance on Ranks with Multiple Comparisons*				
Google Test Case 1	H = 3838.928 with 2 degrees of freedom. (P = <0.001)				
	Comparison	Diff of Ranks	p	q	P<0.05
	R[6] vs S	4083223.000	3	73.663	Yes
	R vs D	488846.000	2	13.228	Yes
	D vs S	3594377.000	2	97.260	Yes
Yahoo Test Case 1	H = 3888.941 with 2 degrees of freedom. (P = <0.001)				
	Comparison	Diff of Ranks	p	q	P<0.05
	R vs S	4097685.000	3	73.924	Yes
	R vs D	524970.000	2	14.205	Yes
	D vs S	3572715.000	2	96.674	Yes
Netscape Test Case 1	H = 4076.532 with 2 degrees of freedom. (P = <0.001)				
	Comparison	Diff of Ranks	p	q	P<0.05
	R vs S	4109911.500	3	74.144	Yes
	R vs D	616623.000	2	16.685	Yes
	D vs S	3493288.500	2	94.525	Yes
Google Test Case 2	H = 1965.204 with 2 degrees of freedom. (P = <0.001)				
	Comparison	Diff of Ranks	p	q	P<0.05
	D vs S	2942150.000	3	53.077	Yes
	D vs R	462700.000	2	12.520	Yes
	R vs S	2479450.000	2	67.092	Yes
Yahoo Test Case 2	H = 2522.305 with 2 degrees of freedom. (P = <0.001)				
	Comparison	Diff of Ranks	p	q	P<0.05
	D vs S	2942150.000	3	53.077	Yes
	D vs R	462700.000	2	12.520	Yes
	R vs S	2479450.000	2	67.092	Yes
Netscape Test Case 2	H = 1878.039 with 2 degrees of freedom. (P = <0.001)				
	Comparison	Diff of Ranks	p	q	P<0.05
	D vs S	2896472.500	3	52.253	Yes
	D vs R	1009745.000	2	27.323	Yes
	R vs S	1886727.500	2	51.053	Yes
Google Test Case 3	H = 1864/994 with 2 degrees of freedom. (P = <0.001)				
	Comparison	Diff of Ranks	p	q	P<0.05
	R vs S	2793600.000	3	50.397	Yes
	R vs D	1543200.000	2	41.758	Yes
	D vs S	1250400.000	2	33.835	Yes
Yahoo Test Case 3	H = 2244.290 with 2 degrees of freedom. (P = <0.001)				
	Comparison	Diff of Ranks	p	q	P<0.05
	R vs S	3160800.000	3	57.022	Yes
	R vs D	1622400.000	2	43.901	Yes
	D vs S	1538400.000	2	41.628	Yes
Netscape Test Case 3	H = 856.910 with 2 degrees of freedom. (P = <0.001)				
	Comparison	Diff of Ranks	p	q	P<0.05
	S vs D	1744800.000	3	31.477	Yes
	S vs R	213600.000	2	5.780	Yes
	R vs D	1531200.000	2	41.443	Yes

[6] S= GA using Structural Genes only, R= GA using Structural and Regulatory Genes without Deletion D=GA using Structural and Regulatory Genes with

Structural GA: The results show that, for Case 1, "purchase, agreement, of", the performance of the Structural GA is not good. In 1600 simulations over three search engines, the fitness values did not improve and resulted in average fitness values of zero. The results are similar for Case 2, which is similar to Case 1, with the added nonsense word "smith". In Case 3, "hotel, place, des, arts", for Google and Yahoo, the results are similar for the Structural GA. However, in Case 3 with Netscape, performance of the Structural GA is better than the other GA conditions. These results suggest that the Structural GA, which uses a homogeneous representation, only performs well in very specific circumstances. It also suggests that Netscape performs differently from the other two search engines. This will be discussed further below.

The fact that the results of Structural GA are very poor is not surprising. Both Case 1 and Case 2 contain general keywords that produce a very large number of returned web pages from search engines. Thus there is little chance that there will be relevant information in the top ten returned web pages, a criteria for a non-zero fitness value. This will result in a fitness value of zero. However, it is clear that Structural GA can function in the right circumstances, as is shown in Case 3 with Netscape where the average fitness value is greater than zero.

Regulatory GA: For Case 1 with all search engines, the Regulatory GA performs very well, reaching average fitness values of 8 or greater. For Case 2, with the addition of the nonsense keyword "smith", performance of the Regulatory GA is reduced and does not perform as well as the Deletion GA with Google and Netscape, but outperforms the other GA conditions for Yahoo. In Case 3, the Regulatory GA outperforms the Deletion GA in all conditions, but performs worse than the Structural GA for Netscape. These results strongly suggest that the heterogeneous representation works well in most conditions.

Deletion GA: For both Case 1 and Case 3, the Deletion GA does not perform as well as the Regulatory GA. However, in Case 2, which contains that silent gene, the Deletion GA performs better than the other GA conditions for two of the three search engines. This suggests that, while the addition of a deletion operator has only a minor impact on

some of the results for test cases without silent gene (particularly Case 1), it works best when a silent gene is present in the representation.

Cases: In general, the average fitness values were highest for Case 1, "purchase, agreement, of" and lowest for Case 2, "purchase, agreement, of, smith". However, the pattern of performance between the different GA conditions was similar in these two cases. The pattern of performance for Case 3, "hotel, place, des arts" was quite different from the other two cases, with Deletion GA performance reduced in comparison with the performance of the Regulatory GA. In addition, the results for Case 3 were significantly different in terms of the variability of the results and terms of the results for one search engine, Netscape.

Search Engines: The pattern of results for Google and Yahoo were very similar for all cases, although the Deletion GA performed slightly better than the Regulatory GA in Case 2 for Google but slightly worse for Yahoo. On the other hand, the pattern of results for Netscape was considerably different than for the other two search engines. This was particularly true for Case 3, where the Structural GA worked better than the other two GA conditions.

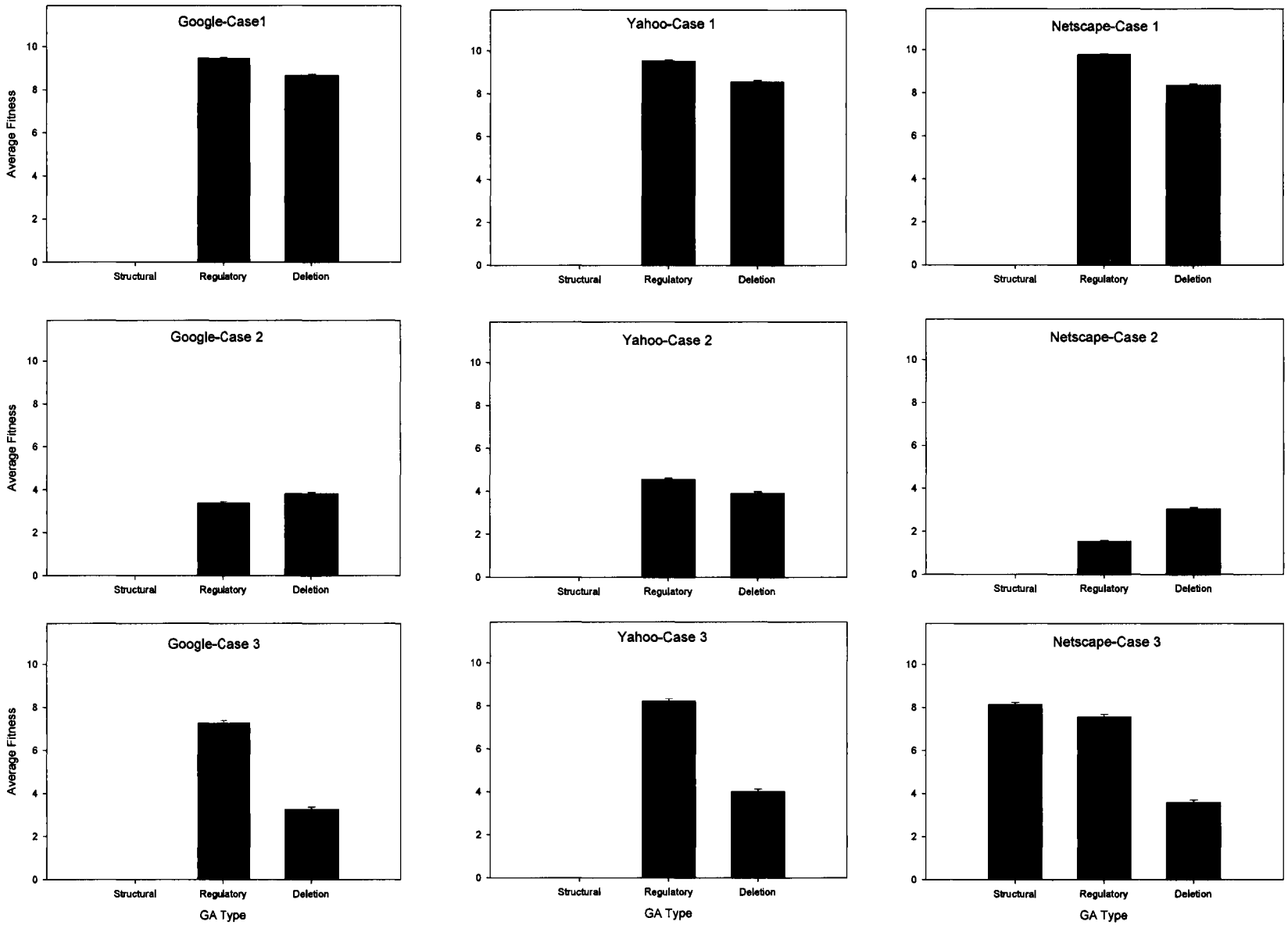

Figure 24: GA performance across all cases (columns) and search engines (rows). Average fitness for the three types of GA shown in each panel along with standard error bars. Parameters are: initial population=20, generations=12 and simulations=1600[7]

[7] All differences in the average values shown in this figure are statistically significant [P<0.05]

DISCUSSION

Summary of Results

The main results of this study are summarized below and then discussed in order of presentation, followed by a discussion and conclusions regarding the impact of this work and future research.

Performance of Simulation Subsystems: Incorporating methods to simulate user behavior and self-recovery as well as using a distributed computational approach to the implementation and an automated fitness ranking algorithm were all successful in making the simulation run efficiently and without interruption when interacting with the Internet.

Performance of Simulation Parameters: The results show that the optimal parameters for the simulation where the largest ones: an initial population of 20, 12 generations and 1600 simulations.

Comparison of GA Performance: All test results show the following:

(1) The heterogeneous representation of the Regulatory GA (Modified GA without deletion) and the Deletion GA (Modified GA with deletion genes) both generally perform better than the Structural GA (Current GA) which uses a homogeneous representation.

(2) The Regulatory GA performs best in the cases without the silent gene

(3) The Deletion GA performs best in the case with the silent gene in the input keywords

(4) The way in which the GA conditions perform varies with the cases. Other than the differences caused by the presence of the silent gene (see (3) above), the results of the GA conditions with Case 1 ("purchase, agreement, of") are very different from those with Case 3 ("hotel, place, des, arts").

(5) The performance of the different versions of GA also varies with the search engines. In particular, Netscape appears to result in a very different performance

from Google and Yahoo. For example, this is the only search engine, which produced a non-zero result for Structural GA.

Impact of Simulation Subsystems and Parameters

In implementing software to conduct experiments and collect empirical data, I came up with different ideas to resolve obstacles that occurred when my programs crawl on Internet to interface to search engines and different web sites. One technique I have developed is designed to simulate a real user interaction to prevent a hostile reaction from search engines and web sites. In addition, programs are designed to be as autonomous and self-recovery as possible so the execution will be continuous within a specific interval of time. This feature allows me to handle the dynamic characteristics of the Web and the web page contents. Finally, distributed instances of the programs executed on multiple computers in parallel is more efficient and allows the completion of a heavy work load with limited resources. These techniques have many potential uses and could be expanded to cope with a large number of keywords and search operators in a more complex implementation of the problem area explored in this study.

The exploration of the optimal simulation parameters indicates that the larger the initial population and number of generations the more likely the GA is to produced optimal fitness. Future work would extend this exploration further to determine the most efficient set of parameters. In the conditions of this study, the parameters did not appear to be affected by the type of query (Case) or the search engine used. However, this could be explored further.

In this study a simulation method was used to allow the efficient study of a number of parameters. However, the next step, would be to attempt to use modified versions of the GA in a real-time, on-line program. This is possible because of the success of the simulation subsystems that have been developed. The automated ranking system can be applied to determine fitness and multiple instances can be run to determine what performance, on average, would be expected.

Comparing the Performance of Structural, Regulatory and Deletion versions of GA

Impact of Modified GA: The test results confirm the validity of my hypothesis; that is, that the modified GA approach, which incorporates additional biological concept, will improve the effectiveness of search engine queries in comparison to the unmodified GA approach. In general the Structural GA, using a homogeneous representation of keywords, was not successful in optimizing queries. The exception to this was in the test case "hotel, place, des, arts" when searching Netscape. This is in part because of Netscape's ranking policy and the placement of paid listings, such as those for hotels and travel destinations. In addition the user ranking rules that were applied in this case in the automatic ranking system were very specific, so that web pages were likely either to fit the ranking rules well or not at all. Netscape appears to be the only search engine that returned enough information to allow the structural GA to evolve.

Impact of Cases: In general, the Regulatory GA was successful at optimizing the queries, although it did perform differently with the different test cases. In the case of the silent gene, it probably performed more poorly due to inability to remove irrelevant web pages. The Deletion GA was more appropriate for this situation. In the case of searching for a hotel near the Place Des Arts, the Regulatory GA was able to improve the fitness, but the variability in GA performance was high. This was due to the large number of instances where the evolution of the query resulted in either a 0 fitness (no appropriate web pages) or a 10 fitness (all web pages appropriate), but nothing in between, as well as the nature of the automatic ranking rules (see above). This is likely due to the nature of the query, which uses a well-known place name. Clearly GA does not work as well in these circumstances and the Structural GA works as well as the more biologically appropriate versions in the case of Netscape.

Impact of Search Engines: The results provide an opportunity to look at the differences among search engines. For example, my results show that Netscape appears to work differently from Google and Yahoo, especially for Case 3. This is a surprising result

since Google and Netscape use the same technology while Yahoo uses a different technology[2]. This difference could be due to the ranking policies and ranking rules. This suggests that the performance of web search engines depend on other factors beside technology. This is an interesting thought as Google technology is currently licensed to other search domains beside Internet search.

Limitations of this research

The research presented here is limited in a number of ways that are outlined below.

Generality of the fitness function: The implementation of the fitness function is not general (i.e., the rules are not applicable for all test cases). This limits the application of this approach in any real-world application. Although this approach was necessary to implement the GA to answer the questions proposed in my research, it limited my ability to compare my results across different queries. Within the scope of this thesis work, the fitness function might not be generalized for all test cases, but it's fitness ranking is applied uniformly.

Objective measures of performance: Due to a nature of problem domain, I was unable to use the more objective measures of precision and recall to evaluate my results. Therefore, it is difficult to compare these results with other techniques in IR. While this would be useful, it was not the primary objective of my work.

Limited parameters: The number of test cases and number of keywords are relatively small for assessing the modifications in terms of how they might work in a real time system. But, given the resource-constrained environment, the variety of test cases used was able to demonstrate some interesting results that also suggest areas for future research.

Comparison to other approaches: Since it is hard to compare GA, especially basic GA with other approaches in general, it is difficult to determine if this approach is the best

one for optimizing web search. However, the goal of this thesis was more limited in scope. I set out to determine if modified GA would improve the result. In future research this would be an interesting area to investigate.

Additional biological concepts: Additional biological concepts such as frame-shift, where individual nucleotides are inserted or deleted could be used, in this case, to model the impact of a misspelled word. Some of these concepts may be more powerful than the simple modifications used here. However, I believe my thesis demonstrates the validity of this approach.

Statistical Analyses: Due to the fact that the data are not normally distributed, there are limits to the complexity of the analyses that can be applied. In future research, this question would have to be addressed.

Implications of Research

In all my experiments, all search engines are black boxes. The same input is used for each search engine and its outputs are collected and analyzed. The way each search engine crawls the web site, extracts and ranks the content is different. Therefore, the same keywords and searched web sites are likely to have different rankings in different search engines. This would result in a kind of performance differences seen in this study.

Also, search engines tune up their searches to appeal to a specific market. For example, Netscape search engine architecture is similar to Google but has a different ranking policy for paid listing. This could be an explanation for why the Netscape search engine behavior is different from Yahoo and Google search engines.

Reverse Engineering: Understanding how search engines use keywords allows web site designers to choose keywords that maximize the probability of getting a higher ranking by a search engine. The work in this thesis is potentially useful in understanding the search engine in order to optimize the web site priority.

Impact on Web Search: The results of this study have a number of implications for web searches and information retrieval from the Web in general. It is a complex process and it's problem domain is fit for GA. Although GA is relatively slow within discrete search space, it is very good approach to deal with very large search spaces such as The Web. By modifying GA to include more appropriate representation based on biological concept, I have demonstrated that this approach is useful.

Almost any one who uses a computer to search the Internet ends up frustrated because they cannot find always what they are looking for. What can we do about it? There are many answers to this question. My work in this thesis shows that it is possible to improve the search performance without requiring changing the search engine architecture. The same approach could be applied to other areas of information retrieval.

CONCLUSION

In my study, I focused on the improvement of search engine performance from the user perspective. By optimizing keywords use, I provide a flexible solution without requiring changes in search engine architecture. This might be the only practical approach now, given the limited choice of search engines in the current Web environment.

In my investigation of method to improve GA for optimizing search queries, I did not anticipate adding to that body of theoretical knowledge. Instead, I attempted to apply the GA approach and theory to a practical problem. The problem is the ability to obtain relevant results from search engines queries using the GA approach. In using biological concepts to solve practical problems, I believe I have added to the theoretical concept by introducing a working model of heterogeneous gene representation and using deletion in combination with silent genes.

This work provides an initial step in a process of applying GA to solve this problem. Online implementation of this process is possible with my approach and is the next logical step in my research.

Another important investigation is to expand the size of query to increase the number of keywords and search operators to determine if this approach will work in the real world. Along with this, it is important to look at the impact of increasing the values of the parameters for initial population and number of generations.

This approach can also be used to understand the behavior of search engines or search algorithms. This would be a reverse engineering approach to the problem of enhancing the ranking of a web site in any search engine.

REFERENCES[8]

[1] Page, Lawrence, and Sergey Brin. "The anatomy of a large-scale hyper-textual web search engine." Proceedings of the Seventh World-Wide Web Conference, 1998, 107-117.

[2] Hedger, Jim. "Technologies at Google and Overture's purchase by Yahoo combine to caused realignments in contextual advertising industry." November 10, 2003, Internet Search Engine Database. August 2005 <http://www.isedb.com/db/articles/540/index.html>.

[3] IBM Research. "Peer to peer networking project introduction." Smartnetworking Project. April 2005 <http://www.research.ibm.com/smartnetwork/peer.html>.

[4] Androutsellis-Theotokis, Stephanos, and Diomidis Spinellis. "A survey of peer-to-peer content distribution technologies" ACM Comput. Surveys 36.4 (2004): 335-371.

[5] Wikipedia. "Peer-to-peer." Wikipedia, the free encyclopedia. February 2006 <http://en.wikipedia.org/wiki/Peer-to-peer>.

[6] Cordon, O., E. Herrera-Viedma, C. Lopez-Pujalte, M. Luque, and C. Zarco. "A review on the application of evolutionary computation to information retrieval" International Journal of Approximate Reasoning 34.2-3 (2003): 121-144.

[7] Wikipedia. "Machine learning." Wikipedia, the free encyclopedia. March 2006 <http://en.wikipedia.org/wiki/Machine_learning>.

[8] Chen, Hsinchun. "Machine learning for information retrieval: Neural networks, symbolic learning, and genetic algorithms." Journal of the American Society for Information Science 31.3 (1995): 194-216.

[9] Dianati, Mehrdad, Insop Song, and Mark Treiber. An Introduction to Genetic Algorithms and Evolution Strategies Waterloo: University of Waterloo, 2002.

[10] Marczyk, Adam. "Genetic Algorithms and Evolutionary Computation." Talks.Origin. March 2006 < http://www.talkorigins.org/faqs/genalg/genalg.html>.

[8] The format of the reference is based on System Science thesis guideline as found: Gibaldi, Joseph. MLA Handbook for Writers of Research Paper 6th ed. New York: The Modern Language Association of America, 2003.

[11] Ursem. Ramus K. "Multinational GAs: Multimodal Optimization Techniques in Dynamic Environments." Proceedings of the Second Genetic and Evolutionary Computation Conference (GECCO-2000), 2000, 19-26.

[12] Holland, John H.. Adaptation in Natural and Artificial Systems: an Introductory Analysis With Applications to Biology, Control, and Artificial Intelligence (Complex Adaptive Systems) 1st ed. Ann Arbor: University of Michigan Press, 1975, 1st MIT Press ed. Cambrige: The MIT Press, 1992.

[13] Griffith, A., J. Millar, D. Suzuki, R. Lewontin and W. Gelbert. An Introduction to Genetic Analysis 6th ed. New York: WH Freeman and Company, New York, 1998.

[14] Bala, J., J. Huang, H. Vafaie, K. DeJong and H. Wechsler. "Hybrid learning using genetic
algorithms and decision trees for pattern classification." Proceedings of the Fourteenth International Joint Conference on Articial Intelligence, 1 (1995), 719-724.

[15] Gordon, Michael D., and Weiguo Fan. "Probabilistic and genetic algorithm for document retrieval." A Communications of the ACM 31 (1988): 1208–1218.

[16] Fan, Weiguo, Michael D. Gordon, and Praveen Pathak. "Personalization of search engine services for effective retrieval and knowledge management." Proceedings of the 2000 International Conference on Information System, 2000, 20-34.

[17] Atlam, El-Sayed, M. Fuketa, K. Morita, and Jun-ichi Aoe. "Documents similarity measurement using field association terms" Information Processing and Management: an International Journal 39 (2003). 809-824.

18 Yang, Jing-Jye, and Robert R. Korfhage. "Query Optimization in Information Retrieval Using Genetic Algorithms: Report on the Experiments of the TREC Project." Proceedings of TREC, 1993, 31-58.

[19] Sullivan, Danny. "Major Search Engines and Directories." Search Engine Watch, April 2004 <http://searchenginewatch.com/links/article.php/2156221>.

[20] Masum, Hassan, Franz Oppacher, and George Carmody. "From genetic to genomic algorithms: steps toward an interdisciplinary synthesis of evolutionary and molecular computation." Carleton Journal Computer Science 6 (2001):17-22.

[21] Obitko, Marek. "Small introduction to Genetic Algorithm." Marek Obitko Web Site, Czech Technical University. April 2003 <http://cs.felk.cvut.cz/~xobitko/ga>.

[22] Meffert, Klaus, Audrius Meskauskas, Jerry Vos, and Neil Rotstan. "Java Genetic Algorithm Package" JGAP. April 2003 <http://jgap.sourceforge.net>.

[23] Agichtein, Eugene, Steve Lawrence, and Luis Gravano. "Learning Search Engine Specific Query Transformations for Question Answering" Proceedings of the Tenth International World Wide Web Conference (WWW-10), 2001, 169-178.

Appendix A

Statistical Analyses of Results of 1600 Simulations with Generations = 12 and Initial Population Size=20 by SigmaSTAT.

S= GA using Structural Genes (Homogeneous Representation of Genes)
R= GA using Structural and Regulatory Genes without Deletion (Heterogeneous Representation of Genes)
D=GA using Structural and Regulatory Genes with Deletion (Heterogeneous Representation of Genes with Deletion)

Google Test Case 1

Normality Test: Failed (P = <0.001)

Kruskal-Wallis One Way Analysis of Variance on Ranks

Data source: Google TC1 in Notebook

Group	N	Missing
S	1600	0
R	1600	0
D	1600	0

Group	Median	25%	75%
S	0.000	0.000	0.000
R	10.000	10.000	10.000
D	10.000	7.000	10.000

H = 3838.928 with 2 degrees of freedom. (P = <0.001)

The differences in the median values among the treatment groups are greater than would be expected by chance; there is a statistically significant difference (P = <0.001)

To isolate the group or groups that differ from the others use a multiple comparison procedure.

All Pairwise Multiple Comparison Procedures (Student-Newman-Keuls Method) :

Comparison	Diff of Ranks	p	q	P<0.05
R vs S	4083223.000	3	73.663	Yes
R vs D	488846.000	2	13.228	Yes
D vs S	3594377.000	2	97.260	Yes

Yahoo Test Case 1

Normality Test: Failed (P = <0.001)

Kruskal-Wallis One Way Analysis of Variance on Ranks

Data source: YTC1 in Notebook

Group	N	Missing
S	1600	0
R	1600	0
D	1600	0

Group	Median	25%	75%
S	0.000	0.000	0.000
R	10.000	10.000	10.000
D	10.000	7.000	10.000

H = 3888.941 with 2 degrees of freedom. (P = <0.001)

The differences in the median values among the treatment groups are greater than would be expected by chance; there is a statistically significant difference (P = <0.001)

To isolate the group or groups that differ from the others use a multiple comparison procedure.

All Pairwise Multiple Comparison Procedures (Student-Newman-Keuls Method) :

Comparison	Diff of Ranks	p	q	P<0.05
R vs S	4097685.000	3	73.924	Yes
R vs D	524970.000	2	14.205	Yes
D vs S	3572715.000	2	96.674	Yes

Netscape Test Case 1

Normality Test: Failed (P = <0.001)

Kruskal-Wallis One Way Analysis of Variance on Ranks

Data source: NTC1 in Notebook

Group	N	Missing
S	1600	0
R	1600	0
D	1600	0

Group	Median	25%	75%
S	0.000	0.000	0.000
R	10.000	10.000	10.000
D	10.000	7.000	10.000

H = 4076.532 with 2 degrees of freedom. (P = <0.001)

The differences in the median values among the treatment groups are greater than would be expected by chance; there is a statistically significant difference (P = <0.001)

To isolate the group or groups that differ from the others use a multiple comparison procedure.

All Pairwise Multiple Comparison Procedures (Student-Newman-Keuls Method) :

Comparison	Diff of Ranks	p	q	P<0.05
R vs S	4109911.500	3	74.144	Yes
R vs D	616623.000	2	16.685	Yes
D vs S	3493288.500	2	94.525	Yes

Google Test Case 2

Normality Test: Failed (P = <0.001)

Kruskal-Wallis One Way Analysis of Variance on Ranks

Data source: GTC2 in Notebook

Group	N	Missing
S	1600	0
R	1600	0
D	1600	0

Group	Median	25%	75%
S	0.000	0.000	0.000
R	3.000	0.000	7.000
D	3.000	3.000	7.000

H = 1965.204 with 2 degrees of freedom. (P = <0.001)

The differences in the median values among the treatment groups are greater than would be expected by chance; there is a statistically significant difference (P = <0.001)

To isolate the group or groups that differ from the others use a multiple comparison procedure.

All Pairwise Multiple Comparison Procedures (Student-Newman-Keuls Method) :

Comparison	Diff of Ranks	p	q	P<0.05
D vs S	2942150.000	3	53.077	Yes
D vs R	462700.000	2	12.520	Yes
R vs S	2479450.000	2	67.092	Yes

Yahoo Test Case 2

Normality Test: Failed (P = <0.001)

Kruskal-Wallis One Way Analysis of Variance on Ranks

Data source: Yahoo TC2 in Notebook

Group	N	Missing
S	1600	0
R	1600	0
D	1600	0

Group	Median	25%	75%
S	0.000	0.000	0.000
R	7.000	3.000	7.000
D	3.000	3.000	7.000

H = 2522.305 with 2 degrees of freedom. (P = <0.001)

The differences in the median values among the treatment groups are greater than would be expected by chance; there is a statistically significant difference (P = <0.001)

To isolate the group or groups that differ from the others use a multiple comparison procedure.

All Pairwise Multiple Comparison Procedures (Student-Newman-Keuls Method) :

Comparison	Diff of Ranks	p	q	P<0.05
R vs S	3341647.500	3	60.284	Yes
R vs D	337695.000	2	9.138	Yes
D vs S	3003952.500	2	81.284	Yes

Netscape Test Case 2

Normality Test: Failed (P = <0.001)

Kruskal-Wallis One Way Analysis of Variance on Ranks

Data source: Data 6 in Notebook

Group	N	Missing
S	1600	0
R	1600	0
D	1600	0

Group	Median	25%	75%
S	0.000	0.000	0.000
R	3.000	0.000	3.000
D	3.000	0.000	3.000

H = 1878.039 with 2 degrees of freedom. (P = <0.001)

The differences in the median values among the treatment groups are greater than would be expected by chance; there is a statistically significant difference (P = <0.001)

To isolate the group or groups that differ from the others use a multiple comparison procedure.

All Pairwise Multiple Comparison Procedures (Student-Newman-Keuls Method) :

Comparison	Diff of Ranks	p	q	P<0.05
D vs S	2896472.500	3	52.253	Yes
D vs R	1009745.000	2	27.323	Yes
R vs S	1886727.500	2	51.053	Yes

Google Test Case 3

Normality Test: Failed (P = <0.001)

Kruskal-Wallis One Way Analysis of Variance on Ranks

Data source: GTC3 in Notebook

Group	N	Missing
S	1600	0
R	1600	0
D	1600	0

Group	Median	25%	75%
S	0.000	0.000	0.000
R	10.000	0.000	10.000
D	0.000	0.000	10.000

H = 1866.482 with 2 degrees of freedom. (P = <0.001)

The differences in the median values among the treatment groups are greater than would be expected by chance; there is a statistically significant difference (P = <0.001)

To isolate the group or groups that differ from the others use a multiple comparison procedure.

All Pairwise Multiple Comparison Procedures (Student-Newman-Keuls Method) :

Comparison	Diff of Ranks	p	q	P<0.05
R vs S	2793600.000	3	50.397	Yes
R vs D	1543200.000	2	41.758	Yes
D vs S	1250400.000	2	33.835	Yes

Yahoo Test Case 3

Normality Test: Failed (P = <0.001)

Kruskal-Wallis One Way Analysis of Variance on Ranks

Data source: Data 8 in Notebook

Group	N	Missing
S	1600	0
R	1600	0
D	1600	0

Group	Median	25%	75%
S	0.000	0.000	0.000
R	10.000	10.000	10.000
D	0.000	0.000	10.000

H = 2244.290 with 2 degrees of freedom. (P = <0.001)

The differences in the median values among the treatment groups are greater than would be expected by chance; there is a statistically significant difference (P = <0.001)

To isolate the group or groups that differ from the others use a multiple comparison procedure.

All Pairwise Multiple Comparison Procedures (Student-Newman-Keuls Method) :

Comparison	Diff of Ranks	p	q	P<0.05
R vs S	3160800.000	3	57.022	Yes
R vs D	1622400.000	2	43.901	Yes
D vs S	1538400.000	2	41.628	Yes

Netscape Test Case 3

Normality Test: Failed (P = <0.001)

Kruskal-Wallis One Way Analysis of Variance on Ranks

Data source: NTC3 in Notebook

Group	N	Missing
S	1600	0
R	1600	0
D	1600	0

Group	Median	25%	75%
S	10.000	10.000	10.000
R	10.000	10.000	10.000
D	0.000	0.000	10.000

H = 856.910 with 2 degrees of freedom. (P = <0.001)

The differences in the median values among the treatment groups are greater than would be expected by chance; there is a statistically significant difference (P = <0.001)

To isolate the group or groups that differ from the others use a multiple comparison procedure.

All Pairwise Multiple Comparison Procedures (Student-Newman-Keuls Method) :

Comparison	Diff of Ranks	p	q	P<0.05
S vs D	1744800.000	3	31.477	Yes
S vs R	213600.000	2	5.780	Yes
R vs D	1531200.000	2	41.433	Yes