

EVOLUTIONARY COMPUTATION IN THE WORLD WIDE WEB

by Hoang Duy Pham

supervised by Dr. Vic Ceiseilski

A dissertation submitted in partial
fulfillment of the requirements for the
degree of Master of Technology
(Computing)

DEPARTMENT OF COMPUTER SCIENCE
RMIT

DECLARATION

I certify that all work on this dissertation was carried out between July 2000 and October 2000 and it has not been submitted for any academic award at any other college, institute or university. The work presented was carried out under the supervisor Dr. **Vic Ciesielski** who proposed the problem and who guided me to solve the problem. All other work in the thesis is my own except where acknowledged in the text.

Signed,

Hoang Duy Pham

30th of October 2000.

ACKNOWLEDGMENT

I would like to thank to my supervisor Dr. Vic Ciesielski for his guidance, suggestion and problem solving during the course of the dissertation.

I would also like to thank to Alexander M. Rosenberg who helps me to correct my writing.

ABSTRACT

A new computation method known as evolutionary computation arises from applying the concepts in nature, “survival of the fittest”. The new computation method provides an opportunity to tackle many problems in industry. This paper reviews the applications of evolutionary computation in the World Wide Web. Three major kinds of applications have been found. Firstly, the performance of information retrieval is improved by generating automatically a more efficient similarity function or making the search agent more adaptive to the search environment. Secondly, genetic computation is applied to manage traffic on the Web. Reducing unnecessary traffic on the Internet is the goal of an evolutionary model for caching Web data. In addition, genetic algorithms are used to equalize the load among web servers in a cluster. Finally, the genetic computation is used to present the Web page in regard to the amount of information or information categories which users are interested in.

In fact, there are not many applications of genetic computation for the Web. This can result from the computing speed. In addition, the reason is the nature of problem domain, which implies that Web problems are unsuitable to consider as an evolution process.

TABLE OF CONTENT

ABSTRACT	IV
TABLE OF CONTENT	V
LIST OF TABLES	VI
LIST OF FIGURES	VII
1 INTRODUCTION	1
2 EVOLUTIONARY COMPUTATION FOR RETRIEVING INFORMATION	1
2.1 SOLUTION OF W. FAN, M. D. GORDON AND P. PATHAK	2
2.2 SOLUTION OF K. ABE, T. TAKETA, H. NUNOKAWA	3
3 GENETIC ALGORITHMS FOR MANAGING WEB TRAFFIC	8
3.1 EVOLUTIONARY MODEL FOR INTERNET DATA CACHING	8
3.2 GENETIC BASED WEB CLUSTER LOAD BALANCING	10
4 GENETIC ALGORITHMS FOR PRESENTING WEB CONTENT	12
4.1 OPTIMIZING WEB PAGE LAYOUT	12
4.2 LEARNING PROFILES IN BROADCASTING APPLICATIONS ON THE INTERNET	15
5 CONCLUSION	18
REFERENCES	19

LIST OF TABLES

1. Table 2.2.1 Initial search parameters of information retrieval agents	6
2. Table 2.2.2 Search parameters at generation 20 in host A	6
3. Table 2.2.3 Search parameters at generation 20 in host C	6
4. Table 3.1.1 Example of chromosome encoding	8
5. Table 3.2.1 Chromosomes example for a pair of K_1 and K_2	11
6. Table 3.2.2 Workload initialization	11
7. Table 4.1.1 Test results	14
8. Table 4.2.1 Experimental settings	16

LIST OF FIGURES

1. Figure 2.2.1 Information retrieval framework	4
2. Figure 2.2.2 Generation mechanism of next information retrieval agent by genetic operators	5
3. Figure 2.2.3 Transition of max, min and average fitness	7
4. Figure 2.2.4 Transition of max and min for MSHN	7
5. Figure 2.2.5 Transition of max and min for MST	7
6. Figure 2.2.6 Transition of max min for MSIN	7
7. Figure 2.2.7 Transition of max min for MANSR	7
8. Figure 3.1.1 Squid cache structure	9
9. Figure 3.1.2 Avg/max fitness over generations	9
10. Figure 3.1.3 Actions over generations	9
11. Figure 3.2.1 Cluster topology with eight servers	11
12. Figure 4.1.1 Overlapping box and penalty values	13
13. Figure 4.2.1 A panel of colours having user selections	15
14. Figure 4.2.2 Coding of an individual	15
15. Figure 4.2.3 Case 1 with (*) and without fitness scaling	17
16. Figure 4.2.4 Case 2 with (*) and without fitness scaling	17
17. Figure 4.2.5 Case 3 with (*) and without fitness scaling	17

1 INTRODUCTION

Darwinian evolution and genetics have given birth to a class of computational methods called evolutionary algorithms, and in particular, genetic algorithms. These evolution strategies provide new opportunities and challenges with increasing applications in industry. This paper reveals the ‘state of the art’ in applications of evolutionary algorithms on the World Wide Web. Three different areas of the World Wide Web have been investigated including retrieving information, managing web traffic and presenting web pages.

Evolutionary algorithms borrow the concept of evolution from nature for their design and implementation. “The evolutionary algorithm process begins by representing solutions as chromosomes. Each solution candidate is then judged in regard to the type of problem to be solved. The quality of solution may be measured by an evaluation function specified in advance or by an interactive subjective selection. Operators, such as conservative operator and innovation operator, are defined to generate variants of rated solutions. The operators are used to consolidate what is already learned by individuals in the population and eliminate weak individuals with respect to their adaptability to the problem”. [BNKF, 1998, p93]

To implement evolutionary algorithms the following needs to be taken into consideration:

- *Problem representation.* The simple traditional genetic algorithm represents the problem’s data as chromosomes, each of which contains a set of genes with each gene corresponding to a binary value (0 or 1). The alternative to this approach is real encoding, which uses real numbers illustrate genes instead of using binary numbers. In addition, a tree structure is usually used in genetic programming. In this structure , leaves contain terminals symbols, which are typically variables or constants, and nodes–function symbols such as square function [Koza, 1992]
- The *selection* procedure is of vital importance to the likely success of genetic approaches. The most common form is roulette wheel. Its principle is that individuals are selected from the population randomly with a probability according to their fitness value measured against the whole population. Other selection methods include tournament selection or rank selection, just to name a few.
- The *crossover* operator exchanges genetic material between selected individuals. The simplest way is, a random position is chosen along the length of the chromosomes of two selected individuals. Then these two individuals swap genetic information from this point onwards creating two new individuals.
- *Mutation* arbitrarily alters one or more genes of a selected chromosome. The aim of this operator is to introduce some extra variability into the population.

2 EVOLUTIONARY COMPUTATION FOR RETRIEVING INFORMATION

The amount of electronic information on World Wide Web is growing dramatically. Consequently, finding right information is a challenge to users, in particular, inexperienced users. In fact, several search engines have been developed to facilitate information retrieval. However, the performances of several popular search engines such as LookSmart, AltaVista and HotBot are rather low in regard to Walker’s study [Walker, 1999]. Many researches have been done to ameliorate this situation. Two approaches to improve the effectiveness of information retrieval by using evolutionary algorithms have been found. The first approach, proposed by Weiguo Fan, Micheal D. Gordon and Praveen Pathak [FGP, 1999], is automatic generation of a matching function, which is used to assess the relevance of found documents. In the second approach [ATN 1999], K. Abe, T. Taketa, H. Nunokawa present a search method whose parameters are automatically adapted to a search environment.

2.1 SOLUTION OF W. FAN, M. D. GORDON AND P. PATHAK

2.1.1 Task description

Search engines can return a number of documents, which are possibly appropriate in responding to search queries according to their collections. The ranks of these documents are calculated by the matching function. It can be concluded that the more accurate the function the better the performance of the search engine. The most common function is the generalized Cosine matching function, which assesses the similarity between query and document vectors [Salton, 1989]:

$$Sim(Q, D) = \frac{\sum_{i=1}^T w_{qi} w_{di}}{Normalizing_factor}$$

where:

Q is query vector.

D is document vector.

T is number of common terms between the query vector and document vector.

w_{qi} is weight factor for the term i in vector Q.

w_{di} is weight factor for the term i in vector D.

Weight values are arbitrarily determined depending on the weighting scheme since there is no formal way to calculate these values. The weighting scheme plays a vital role in improving the search performance. W. Fan, M.D. Gordon and P. Pathak approached the issue by using genetic programming, which automatically generates a matching function, based on relevance feedback from users.

2.1.2 Coding

Each candidate is represented as a tree where leaves are terminal symbols and nodes are functions. The set of parameters for representing candidates to the problem is defined as follows ([FGP, 1999])

Symbols: Terminals: $tf, df, idf, tf_{max}, tf_{avg}, R$

Functions: $+, -, *, /, \log, \sqrt{}$

Term frequency: tf

Document frequency: df

Inverse document frequency: idf

Maximum term frequency: tf_{max}

Average term frequency: tf_{avg}

$$f = 1 - \frac{1}{\frac{\alpha}{P} + \frac{1-\alpha}{R}}$$

The adaptability of each individual is assessed by the fitness function in the expression below:

where

α is defined by users.

P is precision of the retrieval result (ratio of number of relevant documents retrieved to the total number of documents retrieved)

R is recall of the retrieval result (ratio of number of relevant documents retrieved to number of documents retrieved in the whole collection)

As can be seen from the formula, the higher the rank, which users assign to the candidate, the better fitness it has. Consequently, the higher ranked candidate is more likely selected to produce offspring so that what it learned can be transmitted to next generation.

2.1.3 Assessment

The effectiveness of the proposed solution in generating a matching function has been proved in a simulated environment. The interesting point of this approach is the usage of user feedback, which makes the retrieval query more adaptive to user request, and it ameliorates the accuracy of information retrieval. However, the approach does not mention the mutation of genetic programs. It decreases the diversity of population in each generation. In addition, it can reduce the evolutionary capability of system. Another point is timing. Most information retrievals are time constrained. In other words, the searching process should only consume a reasonable amount of time. However, there was no empirical result to assess system performance in terms of time.

2.2 SOLUTION OF K. ABE, T. TAKETA, H. NUNOKAWA

2.2.1 Task description

The authors introduced a new search method, which combines agent technology and evolutionary computation, based on a self-managing information retrieval network. In this network, servers are virtually connected and each server handles its own database and supports the search mechanism on this database. The search agents travel from the user's host to other hosts in the self-managing information network to gather search results and they automatically adapt themselves to the destination host in order to achieve the best possible performance. A list of search agents is managed and automatically updated in each host in the network. The search process is shown in Figure 2.2.1.

For each agent, a search environment is characterized by five different parameters, including:

1. Maximum Search Host Number (MSHN),
2. Maximum Search Item Number (MSIN),
3. Maximum Search Time (MST),
4. Maximum Migration Error Number (MMEN)
5. Maximum Acceptable Number of Search Results (MANSR).

Evolutionary computation is applied to find the most suitable set of these parameters so that the agent can make an efficient search on this host.

2.2.2 Coding

Each individual of the evolution process is composed of a set of the five search parameters mentioned above and is encoded as a binary string, for example,

(MSHN=2), (MSIN=32), (MST=64), (MMEN=3) (MANSR=3) \Rightarrow
 0010|000100000|001000000|0011|0011

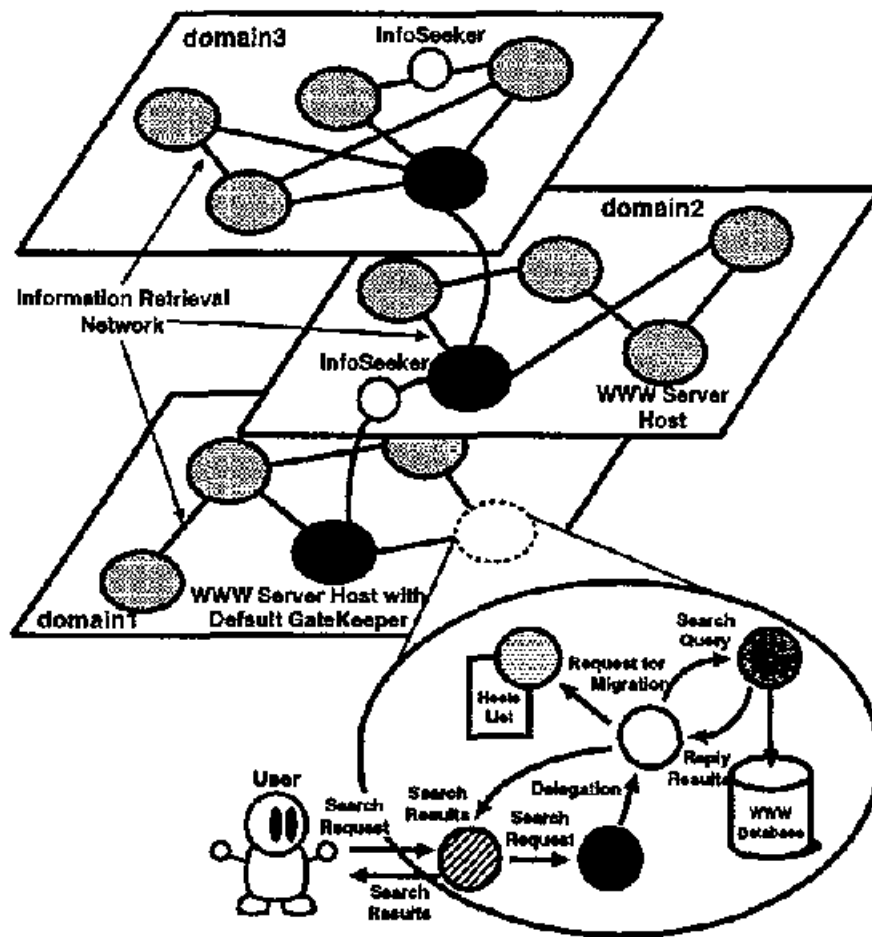


Figure2.2.1 Information retrieval framework [ATN 1999,p.523]

The fitness F of each individual is calculated by the following expression:

$$F = \left(\frac{SH}{MH} + \frac{SI}{MI} \right) \times \left(1 - \frac{ST}{TL} \right) + \left(1 - \frac{ME}{MM} \right)$$

where:

- SH is Search Host Number
- MH is Maximum Search Host Number
- SI is Search Item Number
- MI is Maximum Search Item Number
- ST is Search Time
- TL is Maximum Search Time
- ME is Migration Error Number
- MM is Maximum Migration Error Number

As can be seen from the formula, the quality of search, duration of processing the query and number of network errors are taken into consideration in calculation of the fitness function.

2.2.3 Evolutionary operators

After ranking based on the fitness values, the best individuals are selected to the mating pool. The features of these individuals are transmitted to offspring by using uniform crossover. Instead of exchanging every single bit, whole genes are swapped with respect to the mask pattern.

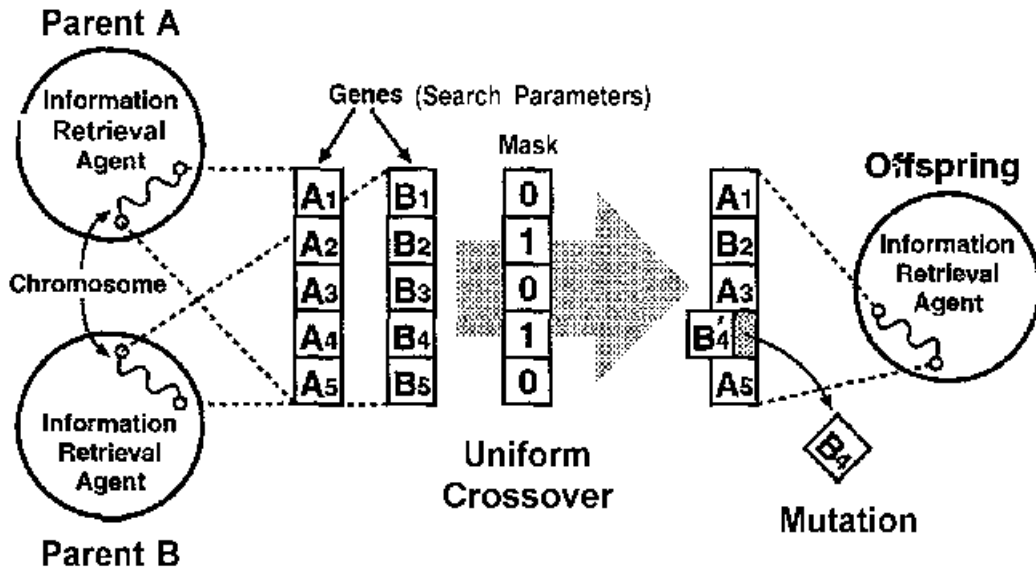


Figure 2.2.2 Generation mechanism of next information retrieval agent by genetic operators [ATN 1999,p.525]

2.2.4 Empirical Result

Three simulated networks of 3, 6 and 9 hosts are created for testing. There are 10 search agents for each host. After every 6 times that a host dispatches a search agent, a new generation is produced and the maximum number of generations is limited to 20. The results are collected during 14,000 migration searches.

The initial search parameters are shown in Table 2.2.1 (MA: Maximum Acceptable number of search results). The sets of parameters for agents in host A and C are in Tables 2.2.2-3. Figures 2.2.5-7 illustrate the changes of maximum search host number, maximum search time, maximum search item number and maximum acceptable search results. The mutation values are denoted as bold characters in the tables.

As can be seen from tables, the values of search parameters in Table 2.2.2-3 are significantly changed compared with initial values but these parameters focus around the same value. In other words, the search agents have adapted themselves to the search environment in each host.

In general, the values generated by the genetic process are not always the best values. These values can vary from generation to generation but during evolutionary process, as shown in Figure 2.23, the fitness has an upward trend. This implies that the more generations produced the better the adaptation.

AN	MH	MI	TL	MM	MA
1	9	85	288	2	8
2	6	26	73	3	3
3	9	88	93	3	4
4	3	77	223	1	5
5	5	29	116	2	5
6	9	46	121	4	5
7	7	51	123	1	3
8	3	54	185	2	9
9	2	69	173	1	9
10	6	33	43	1	2

Table 2.2.1 Initial search parameters of information retrieval agents [ATN

1999,p.525]

the case of 3 host					
AN	MH	MI	TL	MM	MA
1-8, 10	6	29	43	2	2
9	6	63	43	2	2
the case of 6 host					
AN	MH	MI	TL	MM	MA
1-10	3	20	43	4	2
the case of 9 host					
AN	MH	MI	TL	MM	MA
1-10	6	26	43	3	2

Table 2.2.2 Search parameters at generation 20 in host A [ATN 1999,p.527]

the case of 3 host					
AN	MH	MI	TL	MM	MA
1-10	4	26	73	3	3
the case of 6 host					
AN	MH	MI	TL	MM	MA
1-5, 7-10	2	26	178	4	9
6	2	26	178	2	9
the case of 9 host					
AN	MH	MI	TL	MM	MA
1-8	2	26	178	3	9
9	2	52	178	3	9
10	2	26	178	2	9

Table 2.2.3 Search parameters at generation 20 in host C [ATN 1999,p527]

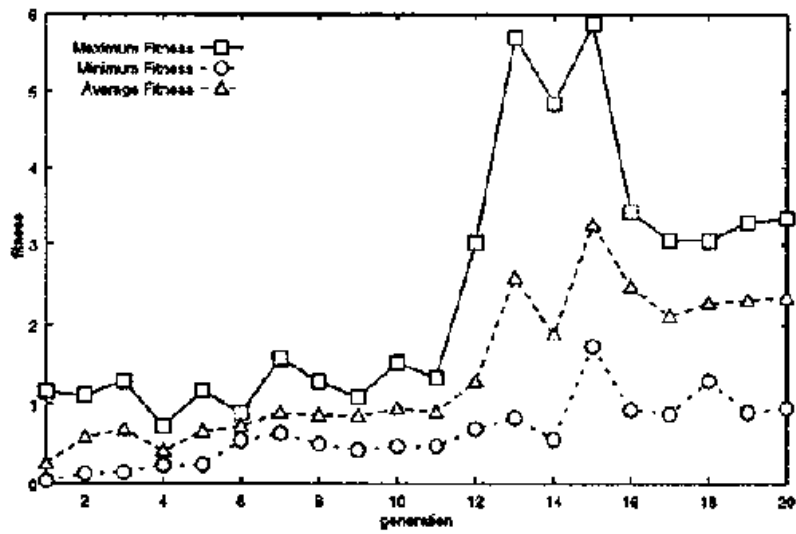


Figure 2.2.3 Transition of max, min and average fitness [ATN 1999,p.527]

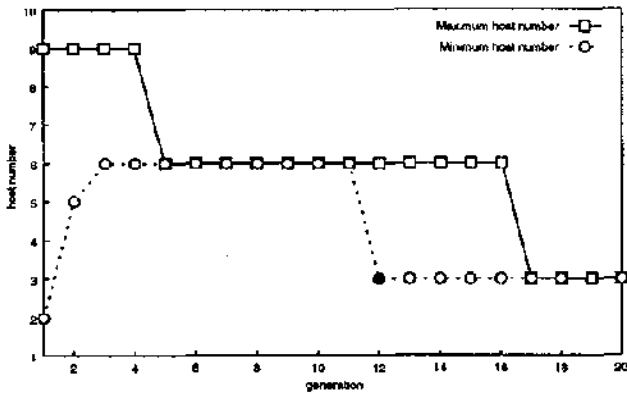


Figure 2.2.4 Transition of max and min for MSHN [ATN 1999,p.526]

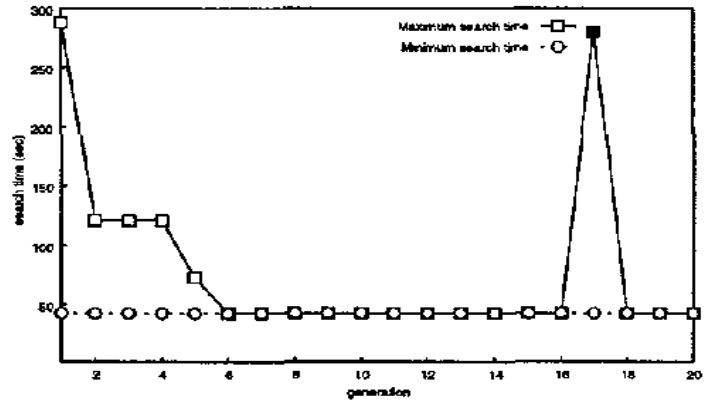


Figure 2.2.5 Transition of max and min for MST [ATN 1999,p.526]

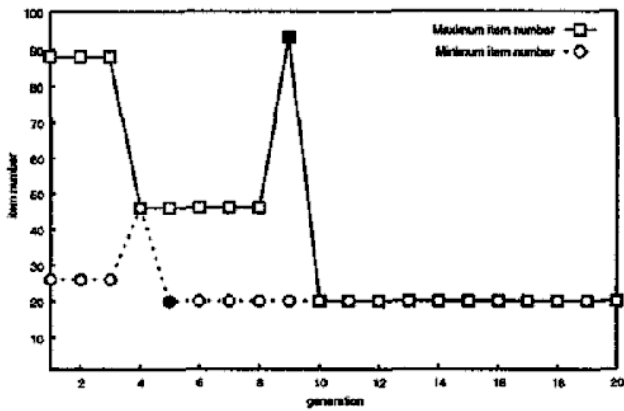


Figure 2.2.6 Transition of max min for MSIN [ATN 1999,p.526]

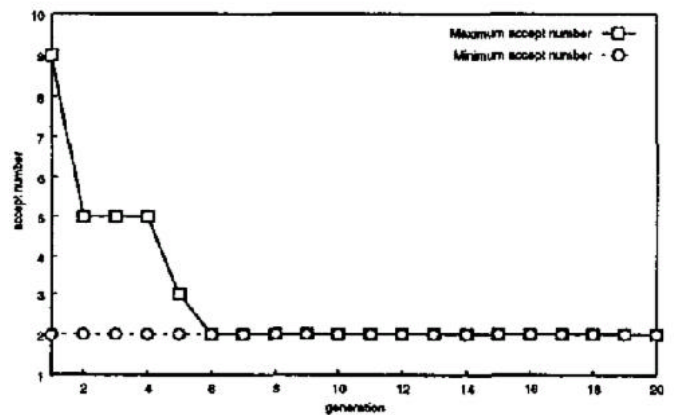


Figure 2.2.7 Transition of max min for MANSR [ATN 1999,p.526]

2.2.5 Assessment

The new search method based on genetic algorithm has proven its effectiveness in the simulated environment. The search parameters converged after a few generations (typically 10). This results from the selection of the fitness function. Another point is that the fitness function reaches a peak at generation 12 and 16 and in the next generations, the fitness goes down. However, the proposed system does not record this state in order to obtain the best generation after evolving.

The performance for each host is not considered, in this search method. This can be the weak point of the solution since the performance of the search mechanism in hosts has a strong influence on the overall accuracy of search results. Therefore, this method should be combined with the solution in part 2.1 so that the search performance can be improved for migration searches and local searches.

3 GENETIC ALGORITHMS FOR MANAGING WEB TRAFFIC

The very large number of users spanning the Internet produces a vast amount of information flows circulating between networks. This gives a challenge to Internet service providers in regard to guaranteeing the quality of service. The next section presents an evolutionary model for Internet caching, which reduces web traffic to the Internet, and web load balancing using genetic algorithms.

3.1 EVOLUTIONARY MODEL FOR INTERNET DATA CACHING

3.1.1 Task description

In practice, the bandwidth of communication channels is limited while the volume of requests from users increase dramatically. Caching is considered as an appropriate solution to highly utilize the bandwidth while providing a high quality of service. The most vital task of the caching technique is that the cache content must reflect accurately the outside of the local network as much as possible. In other words, the degree of the cache consistency determines the level of the success of caching technique. Vakali introduced a model [Vakali 1999], which preserves the consistency of the Web cache by using genetic algorithms.

3.1.2 Coding

According to Vakali, the cache content is composed of files, which are copies of files on the Web. For each user request, firstly, Web servers check the existence of files involving in this request in the cache before retrieving them. Cache is considered as a population of individuals, which are actual cached objects (stored files). The name of each cached object is mapped to a hexadecimal string. This representation is used to encode the chromosomes for each individual.

Name of cached object	Chromosome for the individual
1A5A500	0001 1010 0101 1010 0101 0000 0000

Table 3.1.1 Example of chromosome encoding

The fitness of each individual, which represents the “freshness” of the cached object, is calculated in the following expression

$$fitness_{object} = age1/age2.$$

where:

$age1 = current\ time - retrieval\ time.$

$age2 = retrieval\ time - last\ modified\ time.$

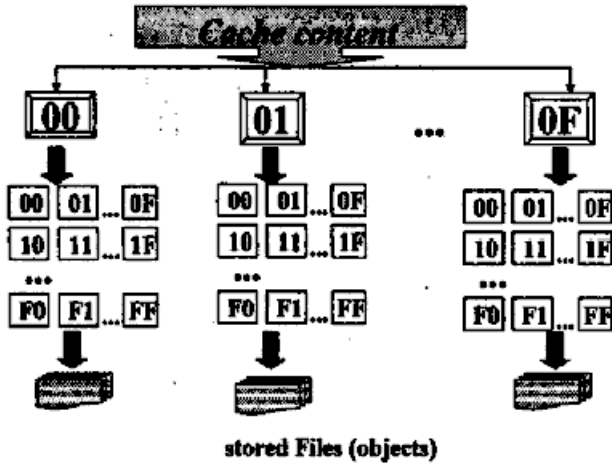


Figure 3.1.1 Squid cache structure [Vakali 1999]

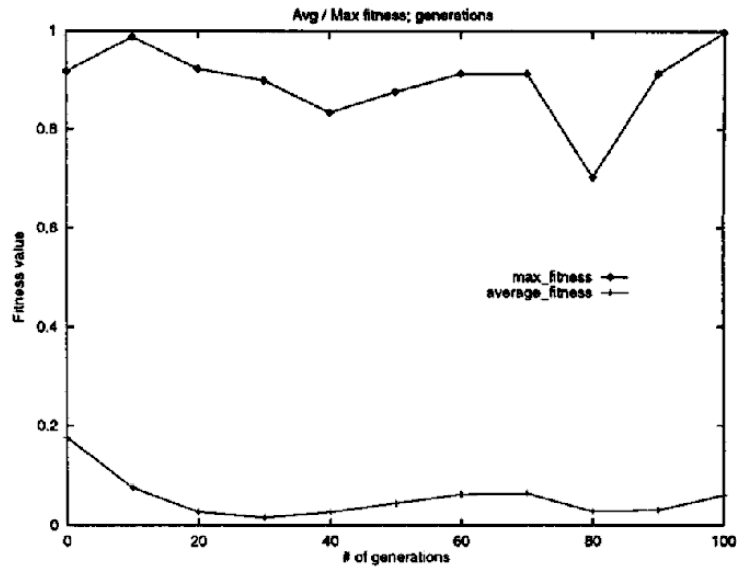


Figure 3.1.2 Avg/max fitness over generations [Vakali 1999,p.653]

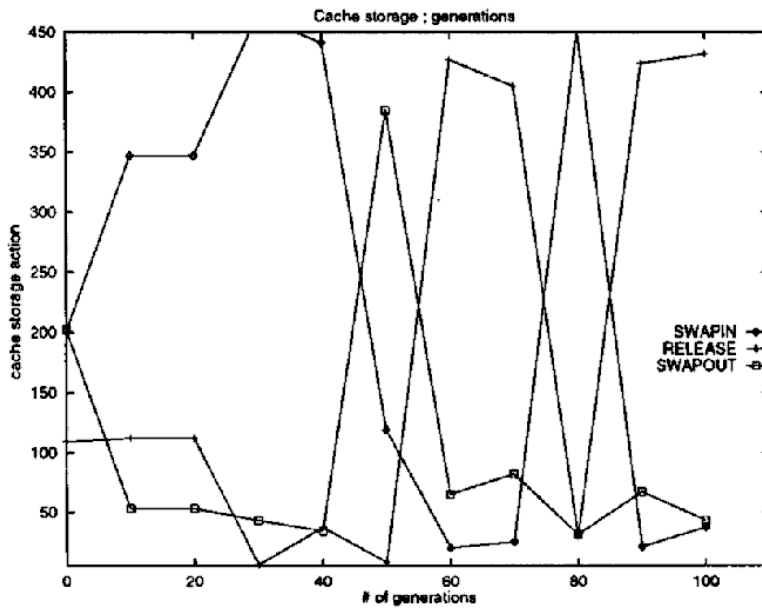


Figure 3.1.3 Actions over generations [Vakali 1999]

3.1.3 Empirical result

The proposed model is embedded in the Squid cache, which was developed by the National Laboratory for Advanced Networking Research, for assessment. The test process lasted three months from October to December 1998. The Squid cache traces and their corresponding log files are analyzed to assess the effectiveness of proposed model.

The initial population is taken from the Squid cache. Crossover and mutation probabilities are 0.6 and 0.0333 relatively. The evolution stops after 100 consecutive generations.

Figure 3.1.1 illustrates the structure of the Squid cache. The transitions of average and maximum of fitness values presents in Figure 3.1.2. According to Figure 3.1.2, the average value of fitness function slightly decreases over generations while the maximum value fluctuates around 0.9. The cache actions at each generation (RELEASE: remove cached object from cache content; SWAPIN: object swapped into memory; SWAPOUT: object saved to disk) is shown in Figure 3.1.3. It is noticed that the number of RELEASE actions increases over generations. This agrees with Figure 3.1.2: the average value of fitness is low. In other words, the number of individuals having small fitness is relative higher than among the others in the population.

3.1.4 Assessment

The proposed model has been proven since the cache population evolved over the simulation time for an increasing number of generations [Vakali 1999]. However, the evolution progress is rather limited during the test, as shown in Fig 3.1.2. This results from the method used by author to approach the problem. In fact, the positions of cached objects in the cache content have undergone the evolution process. The judgement of this individual is based on “external” information, which is the log entry associated with the individual, rather than information coded in its chromosome.

In this model, the adaptability of each cached object is only assessed by time it is retrieved and modified. However, the access frequency of cached object is also important and this factor should be considered as one of major contributors to the fitness.

3.2 GENETIC BASED WEB CLUSTER LOAD BALANCING

3.2.1 Task description

Server clustering is a quite popular technique to provide services continuously to user. In practice, the workload distribution among servers in cluster is not always equal. Cheong and Ramachandran introduced a dynamic workload balancing method, based on Generalized Dimension Exchange, where genetic algorithms are used to optimize the allocation of workload. This method is composed of three steps. First, the generalized dimension exchange algorithm determines a pair of web servers according to their workload by applying edge colouring. Then, the load of each server is modeled by the fuzzy method. The genetic algorithms optimize allocations of the workload of every pair of servers. The process stops if the system reaches to a steady state [ChinRam 2000]. Due to the scope of the paper, only the genetic algorithms step is presented.

3.2.2 Coding

Binary strings are used to represent the tentative solutions. Each bit in the strings corresponds to the state of the server: one illustrates a load (f) in a server (K_i), otherwise–zero. Only one load exists once among the servers at any time. Therefore, the length of the chromosome depends on the number of workloads existing in the cluster.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
K_1	0	0	1	1	0	1	0	0	0	1	1	0	1	0	0
K_2	1	1	0	0	0	0	0	0	1	0	0	0	0	0	1

Table 3.2.1 Chromosomes example for a pair of K_1 and K_2

The fitness of each individual in the population is assessed based on three factors: average size of files, number in the queue and communication cost. The expression as follows:

$$f = 1/\omega_{size} \omega_{number} \omega_{com}$$

where:

ω_{size} is average file size.

ω_{number} is number request in queue

ω_{com} is communication cost

3.2.3 Evolutionary operators

The probabilities of selecting an individual depend on their contributions to the total fitness of the population. In other words, the more the fitness the more likely to be chosen.

Every bit along the chromosomes of a pair of selected individuals has a chance (which is determined by a pre-selected probability) to be interchanged with its counterpart. Therefore, the feature of each chromosome, which has undergone crossover, can be combined regardless of the relative positions of the crossover.

The evolutionary process stops when the fitness function goes over the predefined threshold or the number of generations reaches the threshold, in which the fittest chromosome is taken.

3.2.4 Test procedure

$i(f_i)$	Size(MB)	$i(f_i)$	Size(MB)
1	3.56	11	2.78
2	3.21	12	4.88
3	2.51	13	0.65
4	4.88	14	4.22
5	3.50	15	3.21
6	3.56	16	4.55
7	0.80	17	2.20
8	1.20	18	4.33
9	3.22	19	1.11
10	0.98	20	4.30

Table 3.2.2 Workload initialization [ChinRam 2000,p.718]

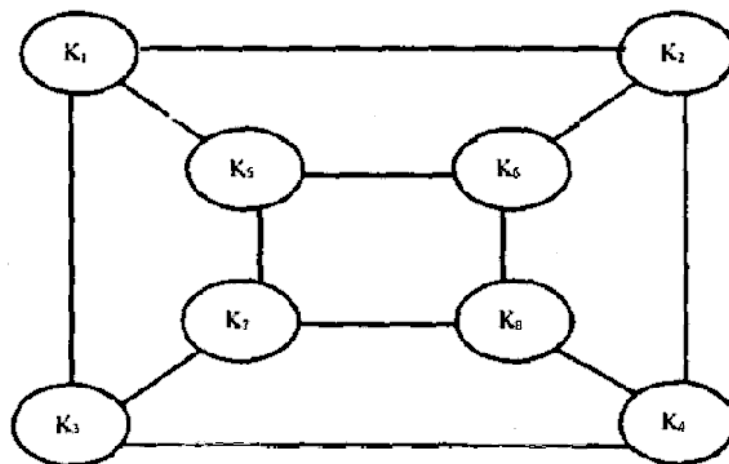


Figure 3.2.1 Cluster topology with eight servers [ChinRam 2000]

The topology of the Web cluster is shown in Figure 3.2.1. There are 8 Web servers denoted by K_1 – K_8 . The probabilities of crossover and mutation are set to 0.5 and 0.1 respectively. The number of crossover points is limited to one. The threshold, which the fitness function has to fulfil, is 0.3. Only the fittest individual is selected after the evolution process.

The evolution process terminates once it achieves a steady state after 100 consecutive generations. Since genetic algorithms is one process in the proposed system, the author provided a limited result of the test.

3.2.5 Assessment

The detail result of running genetic algorithms has not been provided with the article. Based on the set of initial running parameters, it can be deduced that the evolution process costs a quite significant amount of time—the threshold of fitness value and the number of generation are only 0.3 and 100 respectively. If the topology of the cluster becomes more complicated, the set of running parameters has to be carefully chosen.

4 GENETIC ALGORITHMS FOR PRESENTING WEB CONTENT

4.1 OPTIMIZING THE WEB PAGE LAYOUT

4.1.1 Task description

With the support of search engines and the high availability of electric information sources in the Web World Wide, users can easily gain a large number of documents that they are interested in. In fact, browsers have a limited screen surface to represent this information. This section introduces a method to optimize the representation of electronic newspapers, proposed by Penalver and Merelo by using genetic algorithms [PenMer 1998]. The final solutions are refined by using a “greedy” search.

In this method, each electronic newspaper is considered as a rectangular box with fixed size. The browser window has unlimited length and fixed width. A maximum amount of newspapers will be represented without overlap and a minimum surface of the browser is wasted.

4.1.2 Coding

The position of each article box in the browser window is illustrated by X and Y coordinates, which are the coordinates of the upper left of the box. One chromosome is composed of these two values. Due to the limitation of the programming language—chosen by the authors—on bit processing, a pair of integer numbers represents the set of genes in chromosomes rather than a bit string. This affects the implementation of the crossover operator.

The solution is “punished” if it contains overlapping article boxes. The punishment value is calculated based on the width and height of the overlap. The fitness of the solution containing overlaps is reduced. For this special case, the lower the values of punishment and surface used the better the solution. In other words, the higher surface utilization the better.

The fitness of every individual is evaluated by the following expression:

$$F = \left(x + \sum_i^n x_i \right) \left(y + \sum_i^n y_i \right)$$

$x_i = 2s_i, y_i = 0$ if $(x + 2s_i)y > x(y + 2t_i)$
 $y_i = 0, y_i = 2t_i$ otherwise

where:

- n : number of overlapping in this solution.
- x : width of browser screen used by the solution.
- y : height of browser screen used by the solution.
- For each overlapping article i in
 - x_i : horizontal penalty term.
 - y_i : vertical penalty term.
 - t_i : extent of vertical overlap.
 - s_i : extent of horizontal overlap

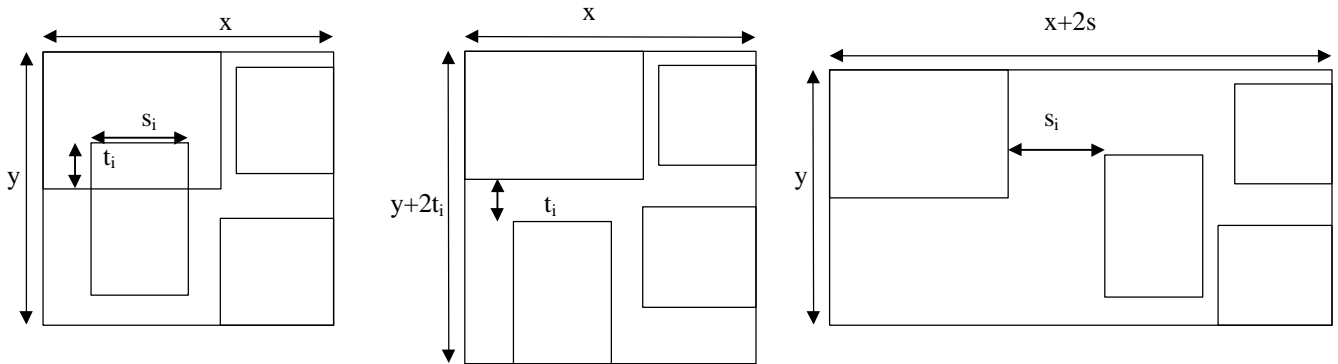


Figure 4.1.1 Overlapping box and penalty values

The size of the overlapping zone is emphasized by doubling the area since this helps the fine tuning process to discriminate good solutions against bad ones (having more overlapping zones).

4.1.3 Evolutionary operators

Two-point crossover is operated on the pair of selected solutions since this method is considered more efficient than one-point crossover [Michalowicz, 1996, p.89]. Moreover, the structure of genes is not destroyed because the crossover operator swaps genes considered as integer numbers rather than binary numbers.

The mutation is conducted to create the diversity of a generation. However, the amplitude of the mutation reduces with time so that the capacity of the system can be finely tuned. The hyperbolic function is used to control the mutation amplitude.

A steady state algorithm is applied in the evolutionary process. This means that the process stops when there are no overlapping article in the solution. In general, fine tuning process, which uses greedy search with heuristic function corresponding to fitness function, starts after 20 generations.

4.1.4 Empirical result

The proposed method has been implemented by using Java script and the program is embedded in the Web page. The test has been conducted on an Intel Pentium II 233 MHz with 64 MB of main memory. The results are collected by reducing the surface of the browser while the number of articles is unchanged such that the ratio of the area used by articles to total surface increases by 10%. Table 4.1.1 presents the percentage of surface usage, the distribution of running time and the distribution of fitness values during 10 executions. The number of population is limited to 50 solutions.

Surface (%)	T \pm σ (ms)	F \pm σ
50	6500 \pm 180	280000 \pm 26000
60	6700 \pm 100	279000 \pm 24000
70	6730 \pm 121	279000 \pm 24000
80	6666 \pm 233	330000 \pm 70000
90	6664 \pm 175	318000 \pm 43000

Table 4.1.1 Test results [PenMer 1998]

As can be seen from the table, the running time (denoted as T) is rather stable with low deviation although the usage of surface increases gradually. Meanwhile, the average surface (denoted as F), used by solutions, increases with high standard deviation (σ) when the available surface reduces.

The authors claimed that with the same quality of final solutions, the greedy algorithm speeds up about five times the overall performance of the system in terms of time, compared with pure genetic algorithms.

4.1.5 Assessment

The proposed method heavily punishes the invalid solutions, which contain overlapping articles, in order to increase the discrimination between best and good solutions [PenMer, 1998] and results in gaps between the article boxes. Further studying on fitness function such as considering gaps as one of major factors in fitness, can eliminate the search algorithm. In addition, the relationship of articles should be taken into account when calculating the fitness of the solution. Overall, the proposed method achieves promising results in regard to the running time and performance of system.

4.2 LEARNING PROFILES IN BROADCASTING APPLICATIONS ON THE INTERNET

4.2.1 Task description

The high diversity of information sources such as news, songs and movies etc. on the Internet gives a challenge for users to locate their favorite information sources. Cuenca and Heudin proposed a system that automatically learns and updates user profiles based on genetic algorithms. Each information source is considered as a channel illustrated by one colour and the system provides a set of channels, which users are probably interested in, in a panel of colours. Then users can tune the channels, by using a number of buttons on the computer screen (see Figure 4.2.1) until they find the channel they want.

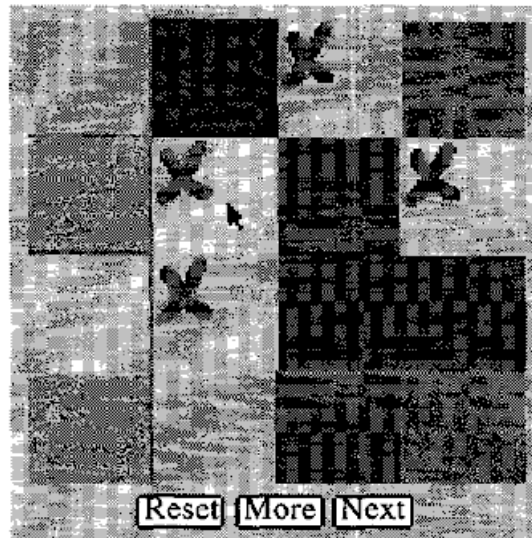


Figure 4.2.1 A panel of colours having user selections
[CuenHeud, 1997]

4.2.2 Coding

The system proposes a panel of 16 colour squares representing 16 different channels. Each colour is encoded by using the RGB format. Therefore, each basic colour (red, green and blue) ranges from zero to 255, and three bytes are enough to code the colour. However, the tentative panel is composed of a string of 32 bit integers since processing an integer number of 32 bits is supported by the programming language. The chromosome for this panel is illustrated in Figure 4.2.2.

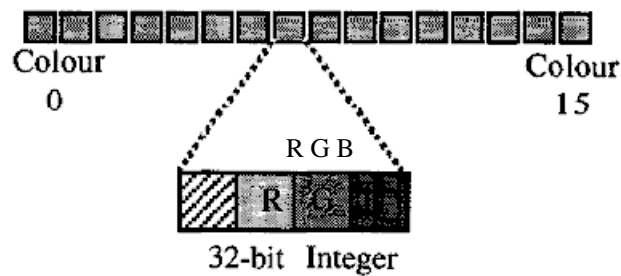


Figure 4.2.2 Coding of an individual
[CuenHeud 1997,p.164]

The fitness of each individual is assessed as follows

$$f(x) = (1 / N) \cdot n_x$$

where N is the number of colour squares in the panel.

n_x is the number of selected colours in the current panel.

4.2.3 Evolutionary operators

The evolutionary process starts by selecting two good individuals (which have high fitness) to undergo mating and two bad ones (which have low fitness), which are replaced by new offspring. The selection operator is based on remainder stochastic sampling without replacement.

The two good ones interchange genetic material by applying one-point crossover with the probability of recombination of 0.7. The diversity of the population is enhanced by using adaptive mutation that can be expressed as follows:

$$x^{t+1} = x^t + N(0, \sigma)$$

where:

x^t is individual undergone mutation.

x^{t+1} is individual after mutation.

N is Gaussian noise function.

σ is standard deviation for that individual.

4.2.4 Empirical Result

The test has been conducted based on three ways that users interact with the system. Users select more than 20% of colours in the panel, less than 20% of colour in the panel and users change choice of colour during the test.

Table 4.2.1 Experimental settings [CuenHeud 1997,p.164]

Number of Colours per Panel	16
Number of Possible Colours	16 777 216
Population Size	64
Expected Number for Selection	2.0
Initial Mutation Probability	0.4
Minimum Mutation Probability	0.03
Crossover Probability	0.7

In the first case, the user selects more than 20% of the colour squares in the panel. In the second case, the user selects less than 20% of the colour squares. In both cases, it is assumed that the users have made a choice of colours they want to see and does not change it during the experiment.

In the third case, the user makes a first choice of colours and suddenly changes to a second set of colours. Figures 4.2.3–4 illustrate the transitions of minimum, average and maximum of the fitness function over generations corresponding to the first and second cases. Figure 4.2.5 presents fitness values when users change their selection at generation 100 of the evolution process. In the first and second case, the upward

trends of fitness are observed. In other words, the proposed system adapts to user behavior over generations.

However, in the third case, the response of the system is slow compared with the first two cases. Despite rather high values of average fitness, the evolution process is low as the fitness function shows a slight increase.

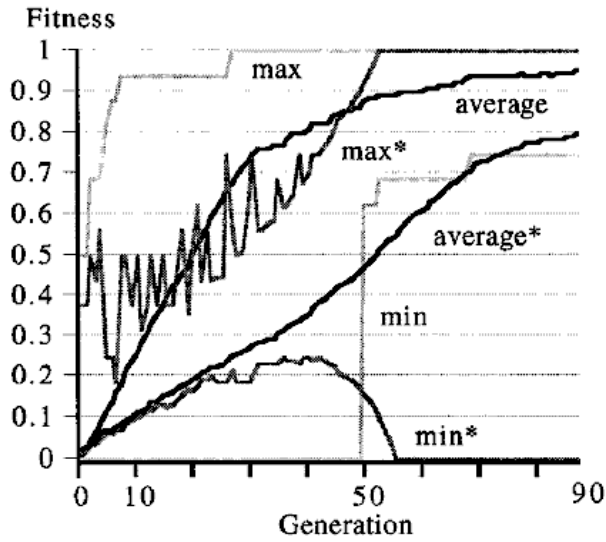


Figure 4.2.3 Case 1 with (*) and without fitness scaling [CuenHeud 1997, p165]

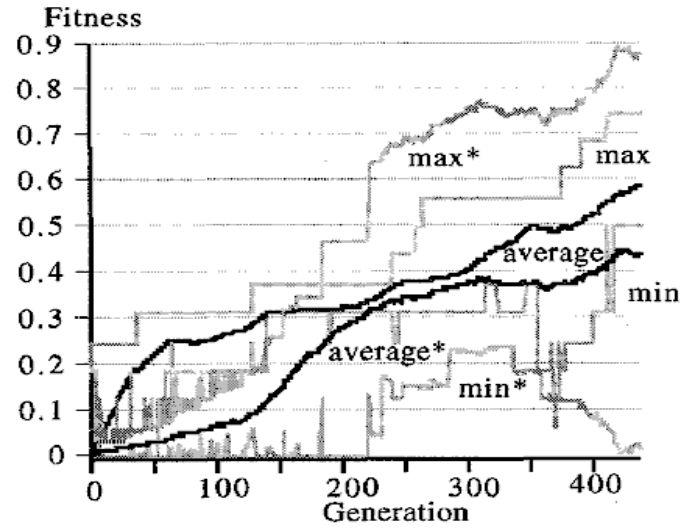


Figure 4.2.4 Case 2 with (*) and without fitness scaling [CuenHeud 1997, p166]

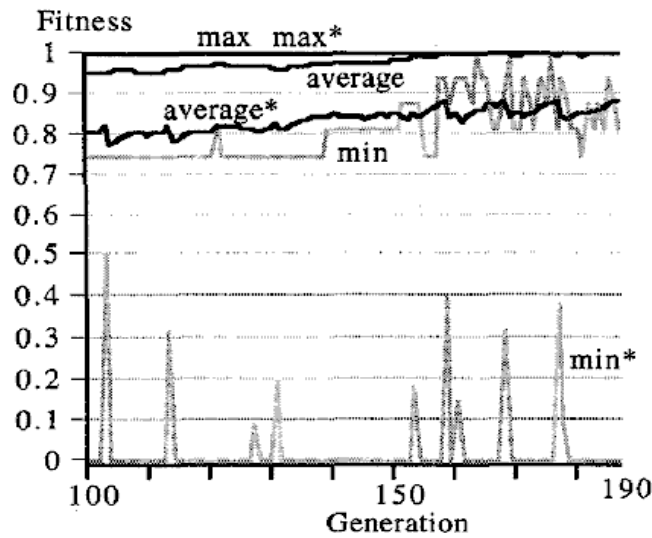


Figure 4.2.5 Case 3 with (*) and without fitness scaling [CuenHeud 1997, p166]

4.2.5 *Assessment*

The system has been proven able to learn the user behaviors. The response of the system is rather good when there are more interactions with users. In other words, the tentative channels proposed by the system are quite close to those that interest users. However, an issue arises when users change their interest. In this case, the system response is not very slow. This can be tackled by using “diploidy”, which represents an individual as a pair of chromosomes, in order to increase the amount of information on problem domain coded in each individual.

5 CONCLUSION

Most of the tasks presented above, especially managing web traffic, are time sensitive. However, these proposed systems have achieved encouraging results in simulated environments. It is persuasive evidence to the applicability of evolutionary algorithms in the World Wide Web.

In fact, there are not many applications of evolutionary computation for the World Wide Web. The reason can be the computation performance, since evolutionary computation consumes a significant amount of computing resources. The other reason can result from the domain of problems. This implies that several problems in the Web are unsuitable to represent as evolutionary processes.

REFERENCES

- [ATN, 1999] K. Abe, T. Taketa, H. Nunokawa, “*An efficient information retrieval method in WWW using genetic algorithms*”. Parallel Processing, 1999. Proceedings. 1999 International Workshops on , 1999 pp: 522 -527
- [BNKF, 1998] W.Banzhaf, P.Nordin, R.E.Keller, Frank.D.Francone. “*Genetic programming : an introduction on automatic evolution of computer programs and its applications*”, Morgan Kaufmann Publishers. Inc 1998 pp91-93
- [ChinRam, 2000] Chin Wen Cheong; V. Ramachandran. “*Genetic based Web cluster dynamic load balancing in fuzzy environment*” High Performance Computing in the Asia-Pacific Region, 2000. Proceedings. The Fourth International Conference/Exhibition on , Volume: 2 , 2000 pp: 714 -719 vol.2
- [CuenHeud, 1997] C. Cuenca and J.C. Heudin “*A genetic algorithm for learning profiles in broacasting applications on the Internet*” Genetic algorithm in engineering system: Innovation and Application, 1997 pp: 163-167.
- [FGP, 1999] W. Fan, M. D. Gordon and P. Pathak “*Automatic Generation of a matching function by genetic programming for effective information retrieval*” Proceedings of the 1999 Americas Conference on Information Systems, pp. 49-51, 13-15 August 1999.
- [Koza, 1992] J. R. Koza “*Genetic programming: On the programming of computers by means of natural selection*” MIT Press Cambrigde MA USA 1992.
- [Michalowicz, 1996] Z. Michalowicz “*Genetic algorithms + data structure = evolution programs*” Third, revised and extended edition Springer–Verlag Berlin Heidelberg New York 1996.
- [PenMer, 1998] J. G. Penalver and J.J.Merelo. “*Optimizing Web page layout Using an annueal Genetic Algorithm as client-side script*” . Parallel Problem Solving from nature—PPSN V, pp. 1018-1027, Springer 1998
- [Salton, 1989] G. Salton “*Automatic text processing*” Addison-Wesley Publishing Co. 1989
- [Vakali, 1999] A. Vakali “*A Web-based evolutionary model for Internet data caching*”. Database and Expert Systems Applications, 1999. Proceedings. Tenth International Workshop on , 1999 pp: 650 -654
- [Walker, 1999] R. L. Walker. “*Assessment of the Web using Genetic Programming*” Proceedings of the Genetic and Evolutionary Computation Conference, Vol. 2, pp. 1750-1755, Morgan Kaufmann, 13-17 July 1999.