Intelligent Spider for Internet Searching

Hsinchun Chen, Yi-Ming Chung, Marshall Ramsey,
Christopher C. Yang, Pai-Chun Ma and Jerome Yen

Abstract— **As the World-Wide Web (WWW) based Internet services become more popular, information overload also becomes a pressing research problem. Difficulities with searching on Internet get worse as the amount of information that available on the internet increases. A scalable approach to support Internet search is critical to the success of Internet services and other current or future National Information Infrastructure (NII) applilcations. A new approach to build intelligent personal spider (agent), which is based on automatic textual analysis of Internet documents, is proposed in this paper. Best first search and genetic algorithm have been tested to develop the intelligent spider. These personal spiders are able to dynamically and intelligently analyze the contents of the users selected homepages as the starting point to search for the most relevant homepages based on the links and indexing.**

An intelligent spider must have the capability to make adjustments according to progress of searching in order to be an intelligent agent. However, the current searching engines do not have the communication between the users and the robots. The spider presented in this paper use Java to develop the user interface such that the users can adjust the control parameters according to the progress and observe the intermediate results. The performances of the genetic algorithm based and best first search based spiders are also reported.

I. Introduction

Information searching over the cyberspace has become more and more important. It has been estimated that the number of homepages is doubled every six months or even shorter. In some areas, such as, Hong Kong and Taiwan, the increasing speeds can be even faster. Searching for the needed homepages or information, therefore, becomes a challenge to the users of Internet. To develop searching engines or spiders which are "intelligent" or have the necessary properties to be called agents is always the dream to the researchers in this area. In order to qualified as an agent or intelligent agent, such searching agent or spider must be able to make adjustments according to progress of searching.

The major problem with the current searching engines is that no communication between the users and the spiders or robots who were dispatched by the users. Since no communication, the users cannot understand the progress of searching and have to tie themselves to the terminals. This paper reports a new searching engines which used Java to develop the user interface. Which allow the users to keep track of the progress of searching, in the mean time, the users can make changes on the searching parameters, such as, the depth and breath, according to the progress reported from the spiders. Besides, various algorithms have been tested for building the spiders, for example, best-first searching and genetic algorithms. The performance of using such algorithms will also be reported in this paper.

Although network protocols and software, such as, HTTP, Netscape and Mosaic, significantly improve the efficiency and effectiveness of importation and fetching of online information. However, their uses are accompanied by the problem that users cannot explore and find what they want in the enormous cyberspace [1], [2], [16]. While Internet services become popular to many users of Internet, difficulties with searching on Internet will get worse as the amount of information stored on the Internet increases. This is mainly due to the problem of information overload [5] and vocabulary differences [8], [4]. We consider that divising a scalable approach to Internet search is critical to the success of Internet services and other current and future National Information Infrastructure (NII) applications.

The main information retrieval mechanisms provided by the prevailing Internet WWW-based software are based on either keyword search (e.g., Lycos server at CMU and Yahoo at Stanford) or hypertext (e.g., NCSA Mosaic and Netscape browser). Keyword search always results in low precision, poor recall, and slow response due to the limitations on in-

MIS Department, Karl Eller Graduate School of Management, University of Arizona, McClelland Hall 430Z, Tucson, Arizona 85721, hchen@bpa.arizona.edu, (520) 621-4153.

MIS Department, Karl Eller Graduate School of Management, University of Arizona, Tucson, Arizona, 85721, ychung@bpa.arizona.edu.

MIS Department, Karl Eller Graduate School of Management, University of Arizona, Tucson, Arizona, 85721, ychung@bpa.arizona.edu.

ECE Department, University of Arizona, Tucson, Arizona, 85721, chrisy@ece.arizona.edu.

Department of ISMT, Hong Kong University of Science and Technology, Clearwater Bay, Kowloon, Hong Kong, pcma@usthk.ust.hk.

Department of Computer Science, The University of Hong Kong, Hong Kong, jyen@cs.hku.hk.

1060-3425/97 $10.00 © 1997 IEEE

dexing and communication (bandwidth), controlled language based interfaces (vocabulary problem), and the inability of searchers to fully articulate their needs. Furthermore, browsing allows users to explore only a very small portion of the large Internet information space. An extensive information space accessed through hypertext-like browsing can also potentially confuse and disorient its users. "embedded digression problem" may cause a user to waste his or her time and learn nothing specific, which is called the "art museum phenomenon" [3].

Our proposed approach, which is based on automatic textual analysis of Internet documents, aims to address the Internet searching problem by creating intelligent personal spider (agent). Best-first search has been tested for local searching, and genetic algorithm has been tested for global searching with subject-specific categories of homepages provided. These personal spiders (agents) can dynamically and intelligently analyze the contents of the users selected homepages as the starting point to search for the most related homepages based on the links and indexing.

II. Literature Review

The WWW was developed initially to support physicists and engineers at CERN, the European Particle Physics Laboratory in Geneva, Switzerland [1]. In 1993, when several browser programs (most noticeably the NCSA Mosaic) became available for distributed multimedia and hypertext-like information fetching, Internet became the preview for a rich and colorful information Cyberspace [17]. However, as the Internet services based on WWW have become more popular, information overload also has become a pressing research problem [2]. The user interactions paradigm on Internet has been shifted from simple hypertext-like *browsing* (human-guided activity exploring the organization and contents of an information space) to content-based *searching* (a process in which the user describes a query and a system locates information that matches the description). Many researchers and practitioners have considered Internet searching to be one of the most challenging and rewarding research areas for future NII applications.

Internet searching has been the hottest topic at recent World-Wide Web Conferences. Two major approaches have been developed and tested: the client-based searching spider (agent) and the online database indexing and searching. However, some systems contain both approaches.

- **Client-based search spiders (agents):**

Broadly defined, an "agent" is a program that can operate autonomously to accomplish unique tasks without direct human supervision (similar to human counterparts such as real estate agents, travel agents, etc.). The basic idea of the agent research is to develop software systems which engage and help all types of end users [15]. Such agents might act as "spiders" on the Internet and look for relevant information [7], schedule meetings on behalf of executives based on their constraints, or filter newsgroup articles based on "induced" (or learned) uses' profiles [11]. Many researchers have focused on developing scripting and interfacing languages for designers and users such that they can create mobile agents for their own [18]. Some researchers attempt to address the question: "How should agents interact with each other to form digital teamwork?". Other researchers are more concerned about designing agents which are "intelligent" [15].

Several software programs based on the concept of spiders, agents, or softbots (software robots) have been developed. TueMosaic and the WebCrawler are two prominent examples. Both of them are using the Best First Search Techniques. DeBra and Post [6] reported tueMosaic v2.42, which developed at the Eindhoven University of Technology (TUE) using the Fish Search algorithm, at the First WWW Conference in Geneva. Using tueMosaic, users can enter keywords, specify the depth and width of search for links contained in the current displayed homepage, and request the spider agent to fetch homepages connected to the current homepage. The Fish Search algorithm is a modified Best First Search. Each URL corresponds to a fish. After the document is retrieved, the fish spawns a number of children (URLs). These URLs are produced depending on whether they are relevant and how many URLs are embedded. The URLs will be removed if no relevant documents are found after following several links. The searches are conducted by keywords, regular expressions, or by relevancy ranking with external filters.

However, potentially relevant homepages that do not connect with the current homepage cannot be retrieved and the search space becomes enormous when the depth and breadth of search become large (an exponential search). The inefficiency and characteristics of the local search of

the BFS/DFS-based spiders as well as the bottleneck of communication bandwidth on Internet severely constrained the usefulness of such agent approach.

At the Second WWW Conference, Pinkerton [14] reported a more efficient spider (crawler). The WebCrawler extends the tueMosaic's concept to initiate the search using its index and to follow links in an intelligent order. It first appeared in April of 1994 and was purchased by America Online in January of 1995. The WebCrawler extended the concept of the Fish Search Algorithm to initiate the search using index, and to follow links in an intelligent order. It evaluates the relevance of the link based on the similarity of the anchor text to the user's query. The anchor text are the words that describe a link to another document. However, these anchor texts are usually short and do not provide relevance information as much as the full document text. Moreover, problems with the local search and communication bottleneck still exist. A more efficient and global Internet searching algorithm is needed to improve the performance of client-based searching agents. Other spiders, such as TkWWW robot, WebAnts, and RBSE (Respository Based Software Engineering), were also developed afterward.

- **Online database indexing and searching:** An alternative approach to Internet resource discovery is based on the database concept of indexing and keyword searching. They retrieve entire Web documents or parts of the documents and store them on the host server. This information is then indexed on the host server to provide a server-based replica of all the information on the Web. This index is used to search for web documents that contain information relevant to a user's query and point the user to those documents. These spiders include World Wide Web Work (WWWW), AilWeb, Harvest information discovery and access system, Lycos, Yahoo, Alta Vista, Excite, etc.

III. Architecture of Intelligent Spider

The architecture of the intelligent spider is divided into five components: Requests and Control Parameters, Graphical User Interface and Client-Server, Search Engine, Indexing Score, and HomepageFetching. Each has different functions. But all the components work together to make the intelligent automatic spider be able to search through the world wide web to find the most related homepages.

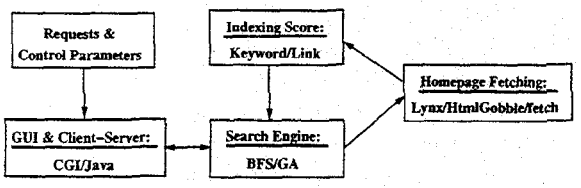

Fig. 1. Architecture of Intelligent Spider

A. Requests and Control Parameters

Tasks are the queries submitted by the users and resources are the databases available on the server. Users submit queries with information, such as, the starting URLs, the keywords, the number of URLs expected to return, and the category of the searching space. A searching will begin whenever a task is submitted. The resources in the server provide several categories of searching space. When a task is submitted, the appropriate searching space in the available database will be invoked.

B. Graphical User Interface and Client-Server

The graphical user interface (GUI) provides a link between the submitted task and the searching engine. It provides graphical interfaces, such as, forms, images, scrollbars, and radio buttons, for the user to fill up the input and control parameters for the searching engine. It also displays the output of the searching engine in the formats of table, graphics, or others. For this project, two interfaces have been developed: Common Gateway Interface (CGI) and JAVA.

The shortcoming of the CGI interface is its lack of dynamic interaction. The server is completely event-driven. It can only responds to requests from the clients, but cannot initiate requests of its own. In addition, it can only respond to one event. These limitations lead to relatively static user interface. The lack of capability in dynamic interaction is undesirable to our spider application. The searching process of the spider usually takes from 5 minutes to 20 minutes to complete. Users are easily frustrated when they are relegated to passive roles during searches and unable to view intermediate results or change parameters in response to searching events. As a result, a truly dynamic user interface which based on a client-server design and TCP/IP is desired. The WWW protocol, HTTP, which is the same protocol used by CGI programs, does not fit this purpose. It is because the two-way data flow cannot be handled

by this protocol.

In order to develop a dynamic interface, the client-server architecture based on the UNIX sockets is created in both C++ and JAVA. JAVA is an object-oriented language for the Internet. It is portable, platform independent, and it creates dynamic applications embedded in HTML documents. The codes for the searching engine are written in C before JAVA is available. Although reproducing the searching engine in JAVA can avoid the communication problems and the programming of sockets, it involves a lot of extra and repeated efforts and the performance may be dramatically reduced. As a result, a client-server architecture based on the UNIX sockets is preferable. Searching is performed by the server that used a slightly modified version of the prototype code. The client was written in Java to enable users to run the user interface on their local computers when they loaded the spider's homepage. The communication between the client and the server is based on the UNIX sockets which allows dynamic client-server communication.

C. Search Engine

For this research, several searching algorithms have been investigated. They include genetic algorithm, best-first search, and simulated annealing. The goals of these searching engines are to visit the URLs in the neighborhood and the URLs in the selected searching space and to find the most related URLs by comparing their links and keywords. The details of these algorithms will be discussed in Section 4.

D. Homepage Fetching

Currently, there are several fetching machines to retrieve HTML documents in the Internet such as Lynx and HtmlGobble. Lynx is a fully-featured WWW client for users running cursor-addressable, character-cell display devices such as vt100 terminals, vt100 emuulators running on PCs or Macs. It will display HTML hypertext documents containing links to files residing on the local system, as well as files residing on remote systems running Gopher, HTTP, FTP, WAIS, and NNTP servers. Similarly, HtmlGobble graps HTML pages from remote web sites. In order to make the spider more portable and lightweighted, instead of using Lynx or HtmlGobble, a generic fetching function is developed to fetch URLs. Using this generic fetching function, it speeds up the fetching time by a significant amount of time. Moreover, the spider code is now more portable and lightweighted.

E. Indexing Score

The goal of indexing is to identify the content of the document (homepage). The indexing method employed here includes the following procedures: word identification, stop-wording, term-pharse formation.

- Word Identification
  Words are identified by ignoring punctuation and case.
- Stop-wording
  A "stop word" list, which includes about 1,000 common function words and "pure" verbs, is developed. The common function words are nonsemantic bearing words, such as on, in, at, this, there, etc. The "pure" verbs are words which are verbs only, such as calculate, articulate, teach, listen, etc. High frequency words that are too general to be useful in representing document content are deleted.
- Term-pharse Formation
  Adjacent words are then used to form pharses. Pharses are limited to three words. The resulting pharses and words are referred as the keywords of the documents (homepages).

After retrieving the keywords of the homepages, the coccurence pattern of indexes which appears in all homepages are identified. Jaccard's score is used to measure the similarity of homepages. The score is computed in terms of the homepages' common links and term frequency and homepage frequency. The detail formulation is available in Section 4.1.

IV. Using AI in Spiders

In this research, we have investigated several AI techniques for developing an intelligent spider (agent) for more efficient and optimal client-based search of relevant Internet information. These techniques include best first search, genetic algorithm, and simulated annealing. Although we have investigated the simulated annealing, it is discared because it does not show significance difference in its performance comparing with genetic algorithm. In this section, the best-first search and genetic algorithm are discussed. The simulated annealing is briefly covered. These techniques have different searching and control mechanisms, but they all compare the similarity of homepages based on the Jaccard's score.

A. Jaccard's Scores

In order to compare the similarity between two homepages, Jaccard's scores are used. In this application, Jaccard's score are computed base on the homepages' links and indexing. A homepage with a higher Jaccard's score has a higher fitness with the input homepages. The Jaccard's scores are computed as follows:

- **Jaccard's Scores from Links**
  Given two homepages, x and y, and their links, X = x_1,x_2, ..., x_m and Y = y_1,y_2, ..., y_n, the Jaccard's score between x and y based on links is computed as follows:

$$\frac{\#(X \cap Y)}{\#(X \cup Y)} \quad (1)$$

  where #(S) indicates the cardinality of set S.
- **Jaccard's Scores from Indexing**
  Given a set of homepages, the terms in these homepages are identified. The term frequency and the homepages frequency for each term in a homepage are then computed. Term frequency, tf_{xj}, represents the number of occurrences of term j in document (homepage) x. Homepage frequency, df_j, represents the number of homepages in a collection of N homepages in which term j occurs.
  The combined weight of term j in homepage x, d_{xj}, is computed as follows:

$$d_{xj} = tf_{xj} \times log(\frac{N}{df_j} \times w_j) \quad (2)$$

  where w_j represents the number of words in term j, and N represents the total number of homepages.
  The Jaccard's between homepage x and y based on indexing is then computed as follows:

$$\frac{\sum_{j=1}^{L} d_{xj} d_{yj}}{\sum_{j=1}^{L} d_{xj}^2 + \sum_{j=1}^{L} dyj^2 + \sum_{j=1}^{L} d_{xj} d_{yj}} \quad (3)$$

  where L is the total number of terms.

B. Searching Methods:

Three algorithms, best-first search, genetic algorithm, and simulated annealing, are investigated. The detail of these algorithms are as follows:

- **Best First Search:**
  Best First Search is a state space search method [13]. It looks for the best homepage at each iteration and the number of iterations equals to the number of output homepages required by users. A sketch of the best first search algorithm is presented below:
  1. **Input Homepages and initialization:**
  Initialize k to 1. A set of homepages, $input_1$, $input_2$, ..., $input_m$, are obtained from users. These input homepages are fetched and their linked homepages are saved in H = { h_1, h_2, ... } .
  2. **Determine the Best homepage:**
  Determine the best homepage in H which has the highest Jaccard's score among all the homepages in H and save it as $output_k$.
  The Jaccard's score based on links for h_i is computed as follows:

$$JS_{links}(h_i) = \frac{1}{N} \sum_{j=1}^{N} JS_{links}(input_j, h_i) \quad (4)$$

  where $JS_{links}(input_j, h_i)$ represents the Jaccard's score between $input_j$ and h_i based on links using Equation (1).
  The Jaccard's score based on indexing for h_i is also computed similarly:

$$JS_{index}(h_i) = \frac{1}{N} \sum_{j=1}^{N} JS_{index}(input_j, h_i) \quad (5)$$

  where $JS_{index}(input_j, h_i)$ represents the Jaccard's score between $input_j$ and h_i based on links using Equation (3).
  The Jaccard's score for h_i is computed as follow:

$$JS(h_i) = \frac{1}{2}(JS_{links}(h_i) + JS_{index}(h_i)) \quad (6)$$

  The homepage in H which have the highest Jaccard's score is saved as an output, $output_k$.
  3. **Fetch the best homepage**
  Fetch the best homepage, $output_k$, and add all its linked homepages to H. Increase k by 1.
  4. **Repeat until all the output homepages are obtained**
  Repeat 2 and 3 until k equals to the total number of outputs plus one required by users.
- **Genetic Algorithm:**
  Genetic algorithms (GAs) [9], [12], [10] are problem solving systems based on principles of evolution and heredity. Genetic algorithms perform a stochastic evolution process toward global optimization through the use of crossover and mutation operators. The search space of the problem

is represented as a collection of individuals which are referred as chromosomes. The genetic algorithm find the chromosomes with the best "genetic material". The quality of a chromosome is measured with an evaluation function (Jaccard's score). In the algorithm, the initial population is chosen and the quality is determined. In every iteration, parents are selected and children are produced by crossover and mutation operations. Each iteration is referred as a generation.

A sketch of the genetic algorithm for Internet client-based searching is presented below:

1. **Initialize the search space**

The spider will attempt to find the most relevant homepages in the search space, using the user-supplied starting homepages. Save all the input homepages, $input_1$, $input_2$, ..., into a set of CurrentGeneration, $CG = \{cg_1, cg_2, \cdots\}$.

Homepages in 14 categories will be used as the search space for mutation. The categorization is done by multi-layered Kohonen self-organizing feature map in our earlier work. These categories includes arts, business, computer, education, entertainment, government, health, news, recreation, reference, regional, science, social-culture, and social-science.

2. **Crossover and Mutation:**

- Crossover

  Fetch the homepages that linked by the homepages in CurrentGeneration. Compare the linked homepages of these fetched homepages and the linked homepages of the homepages in CurrentGeneration, cg_i. The fetched homepages which have the highest number of overlapping linked homepages with the homepages in CurrentGeneration will be saved in the set of CrossoverHomepages, $C = \{c_1, c_2, \cdots\}$.

- Mutation

  Homepages are obtained from the ranked homepages using SWISH in the user selected category and saved in the set of MutationHomepages, $M = \{m_1, m_2, \cdots\}$.

The sizes of CrossoverHomepages and MutationHomepages are both 50% of the population size. The population size, N, is double of the number of output homepages requested.

3. **Stochastic Selection Scheme based on Fitness:**

Compute the Jaccard's score for each homepage in CrossoverHomepages (C) and MuataionHomepages (M) using Equations (4), (5), and (6). The homepages in CrossoverHomepages and Mutation Homepages are compared with the homepages in ElitePool, $E = \{e_1, e_2, \cdots, e_N\}$. The best N homepages are selected and updated to the ElitePool. (For the first generation, ElitePool is empty, comparison is not processed. Insteads, the homepages in CrossoverHomepages and MutationHomepages are saved to the ElitePool and automatically become the best N homepages.) Selection of the population for the next generation is based on the fitness scores. Fitter homepages in the Elitepool have better chances of getting selected. Create a "roulette wheel" with slots (F) sized according to the total fitness of the population. F is defined as follows:

$$F = \sum_{i=1}^{N} JS(e_i) \tag{7}$$

Each homepage in these best homepages has a certain number of slots proportional to its Jaccard's score. A homepage is selected by spinning the wheel and saved to CurrentGeneration. The total number of spinning is N. Some homepages will be selected more than once. This is in accordance with the genetic inheritance: the best chromosomes get more copies, the average stay even, and the worst die off.

4. **Converge:**

Repeat 2 and 3 until the improvement in total fitness between two generations is less than a threshold. The final converged set of homepages is presented as the output homepages.

Spider based on simulated annealing is also developed, however, its performance is not better than genetic algorithm based spider, it is abandoned. Simulated annealing algorithm is based on analogy between the simulation of the annealing of solids and the problem of solving large combinatorial optimization problems. At the first iteration, a high temperature is initialized. In each iteration of simulated annealing, procedures similar to the crossover operation in genetic algorithm are used to produce a new set of URLs (configuration). Jaccard's scores are used as the cost function to evaluate the current configuration. If the Jaccard's score is higher than the score of the previous configuration, the new configuration is accepted, otherwise, it will be accepted based on the probability computed in terms of the current temperature and the Jaccard's score. At the beginning of each iteration, the temperature is decreased. The process terminates when the temperature reaches a

certain value or the change of the Jaccard's score between two iterations is less than a threshold.

Since the simulated annealing based spider only employ the crossover operation to produce new configuration, it is only a local search. However, its performance is worse than best-first search and genetic algorithm. A hybrid technique based on simulated annealing with the addition of mutation operation in producing new configuration is also investigated. Experiments are conducted. Its performance is closer to the genetic algorithm based spider but it is not better. As a result, we abandoned the simulated annealing and the hybrid technique in our development of intelligent spiders. For the rest of the paper, only the performances of best-first search and genetic algorithm are reported.

V. Experiment

In an attempt to examine the quality of results obtained by genetic algorithm and the best-first search, we perform experiments and analyze their results. Experiments are conducted to compare the performance and efficiency of the best-first search and genetic algorithm based spiders. The same sets of input URLs are submitted to both spiders, their results are compared based on the Jaccard's score in Equation (6), and their cpu time and system time taken to process the search are also compared.

A. Experimental Design

In our experiment, 40 cases are set up. For each cases, 1 to 3 input homepages are submitted to the spiders based on best-first search and genetic algorithm. 10 output homepages are obtained as result. Homepages are chosen in the entertainment domain. The average of the output homepages' Jaccard's score for each case is recorded for comparing their fitness. The cpu time and the system time for processing the search on each case are also recorded. In the experiment, we also recorded whether the output URL is origined from mutation. This will give us an idea on the percentage of the output URLs that are contributed by the mutation operation in the global search.

B. Experimental Result

Figure. 2 shows the statistics of the fitness on the results obtained by the best-first search and genetic algorithm performed on the 40 cases. The results show that the output homepages obtained by genetic algorithm has a slightly higher fitness score than those obtained by best-first search, but the difference is not significant. The average of 40 average Jaccard's score for genetic algorithm and best-first search are 0.08705 and 0.08519, respectively. Although the Jaccard's score does not show any significant difference between the performances of genetic algorithm and best-first search, 50% of the homepages that obtained from genetic algorithm are from mutation when the crossover and mutation rate are 50% and 50%, respectively. These homepages that obtained from mutation are most probably not linked to the input homepages, therefore, most of them will never be obtained by best-first search. In particular, when the number of links of the input URLs is restricted, the result of best-first search is poor and the best-first search may not be able to provide as many URLs as requested by the users for result. However, the genetic algorithm does not have this problem because it is a global search, the mutation operation break through this restriction in searching. In the situation of restricted links of the input URLs, the genetic algorithm is still performing very well, a significant difference between their performance is observed.

VI. Current Status

As mentioned earlier, we have developed two interfaces for our spiders. One is based on CGI and the other based on JAVA. The CGI enables image maps and fill-out forms to interact with the http server, unfortunately, it does not provide dynamic interaction during the searching process. On the other hand, JAVA is an object-oriented, platform-independent multi-threaded, dynamic general-purpose programming environment for the Internet, intranet, and any other complex, distributed network. JAVA has the capability to display intermediate results and accept changes of input parameters dynamically. Using the Java interface, users are able to observe the result in every iteration of the searching process, and even adjust the control parameters of the searching mechanism. This dynamic interaction is powerful and users can find what they want more efficiently.

Figure 3 shows the homepage which display the final result of the searching using the CGI based user interface. It displays the total average fitness (based on Jaccard's score) and the number of homepages have been visited in the search. For each homepage, its address, score, title, and matched keywords are also displayed.

Figure 4 shows the window which displays the result of the searching dynamically. The upper left

```
ANALYSIS OF VARIANCE
SOURCE     DF        SS        MS        F        p
FACTOR      1   0.00007   0.00007     0.03    0.857
ERROR      78   0.16535   0.00212
TOTAL      79   0.16541

                                    INDIVIDUAL 95 PCT CI'S FOR MEAN
                                    BASED ON POOLED STDEV
 LEVEL      N      MEAN     STDEV  ----------+---------+---------+------
GA         40   0.08705   0.04176     (-------------*-------------)
BFS        40   0.08519   0.04996   (-------------*-------------)
POOLED STDEV =  0.04604             0.080     0.090     0.100
```

Fig. 2. The statistics of the average Jaccard's scores obtained from 40 cases of searching by genetic algorithm and best-first search.

corner is an animation. A light bulb is fleshing when the process is initialized. During the process, a spider is going up and down to catch a fly when a homepage is being fetched. When the fetching of a homepage is done, the spider will catch the fly. When the searching is done, the spider in the animation will go to sleep. In the upper left corner, there are buttons and scroll bars to control the process dynamically. Similar to the input window, buttons for new search, back to last window and exit are available, scroll bars for controlling the crossover/mutation rate, allowable fetching time, and number of return URLs expected are available. Under the buttons and scroll bars are the text areas to display the status of the searching process. The four text areas on the first row display the total amount of time taken, number of URLs fetched, number of URLs origined from mutation, and the current number of generation according to the genetic algorithm. The text areas in the second row displays the current fetching URL and the amount of time has been used to fetch this URL. It should be less than the allowable fetching time shown on the scroll bar. Below the text areas, there are two text fields. The text fields on the left displays the result URLs in the current generation. Their titles, keywords, and links are also displayed. If the title is clicked, a new browser is opened and the corresponding homepage is fetched. The text fields on the right display the score bars for the corresponding URLs. The first score bar displays the Jaccard's score based on indexing and the second score bar displays the Jaccard's score based on links. The third score bar display the fetching time score. For the titles and the addresses of all the homepages in this window, the color represents whether the homepage is a local URL, non-local or origined form mutation. For example, if the input URL is http://ai.bpa.arizona.edu, http://ai.bpa.arizona.edu/ent is a local URL, http://www.arizona.edu is a non-local URL, and http://www.musenet.org which is origined form the mutation database is a mutated URL.

VII. Conclusion and Discussion

This research presents two algorithms for intelligent personal spiders (agents). The two algorithms are best-first search and genetic algorithm. The best-first search only supports local search, while the genetic algorithm supports global search. The simulated annealing algorithm is discarded in our work because it does not provide a better performance in the experiments. The spiders obtain a set of homepages from users and search for most relevant homepages. They operate autonomously without any human supervision. The results show that genetic algorithm received a higher fitness score although it is not significantly higher, however, it is time consuming. Currently, we are optimizing the algorithm to reduce the cpu time and system time.

Although a significant higher Jaccard's score was not obtained from the genetic algorithm, the users evaluation shows that the subjects agreed that genetic algorithm obtain significantly more relevant homepages. These intelligent spiders are promising in searching for related homepages on the interneton the internet. The genetic algorithm based spider also has made some brokenthroughs in lifting the limitation of the best-first search which was restricted to local search. The best-first search suffers in a poor performance when the number of links from the input homepages is small or when the links of the input URLs are restricted to local site. Using the genetic algorithm, these problems can be partially removed and more potential homepages can be explored.

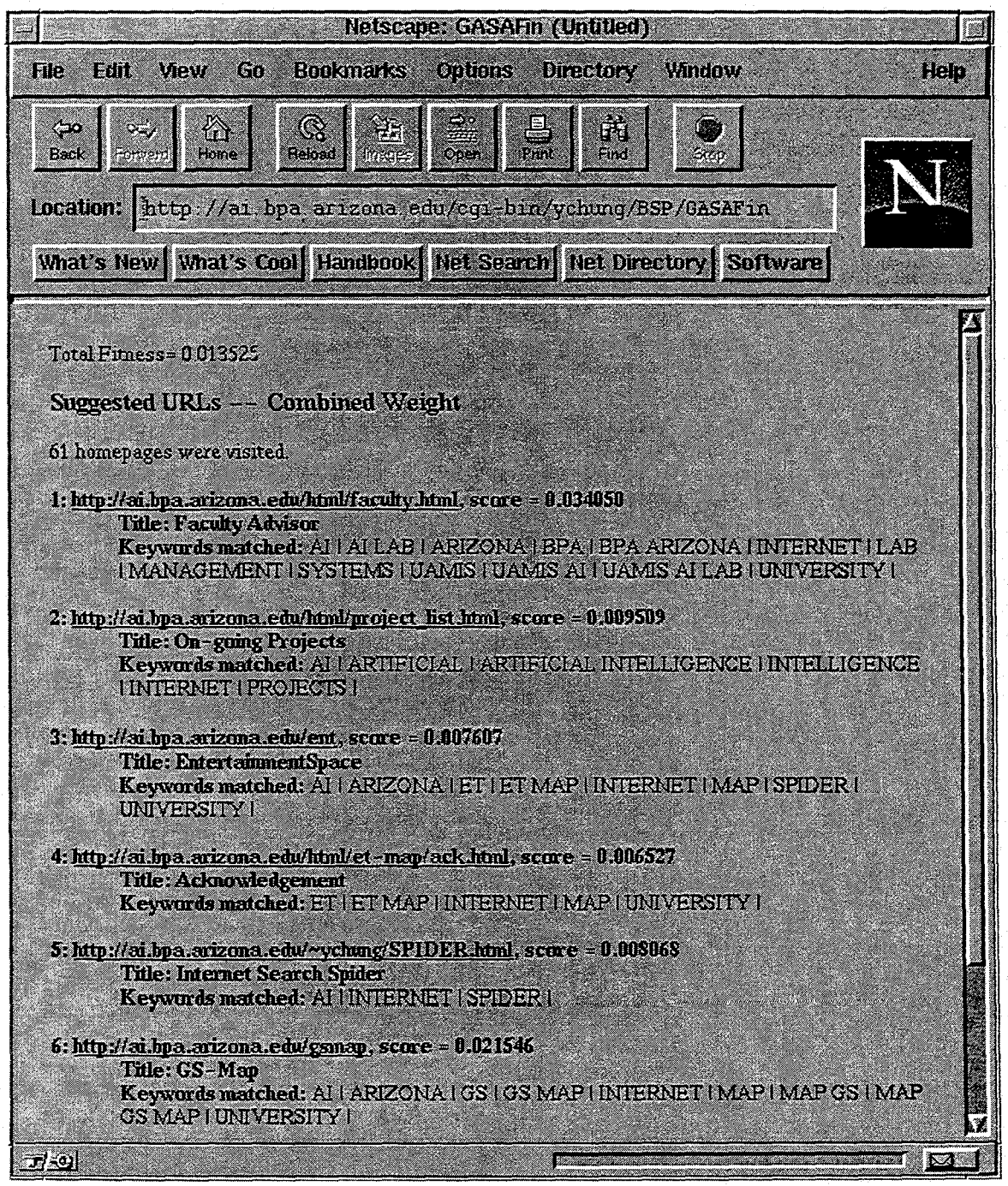

Fig. 3. The output homepage of the global search spider based on genetic algorithm.

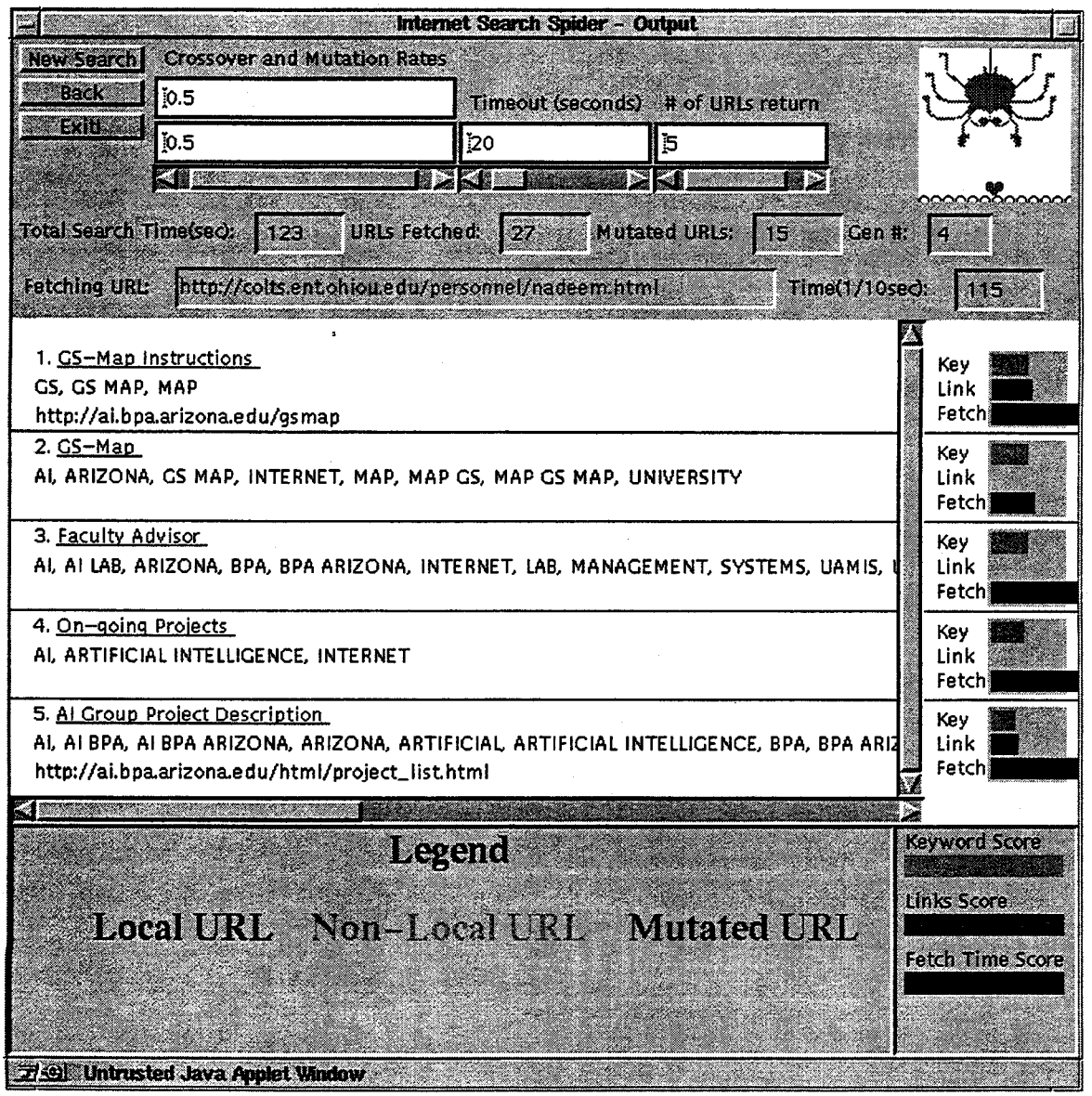

Fig. 4. The window shows the result of the search during the process dynamically. Animation is display on the upper left corner. Control panel which allows user to change parameters during the process is located at the upper left corner. The results are shown at the center of the window.

VIII. Acknowledgment

This project was supported mainly by the following grants:

- NSF/ARPA/NASA Digital Library Initiative, IRI-9411318, 1994-1998 (B. Schatz, H. Chen, et. al, "Building the Interspace: Digital Library Infrastructure for a University Engineering Community"),
- NSF CISE, IRI-9525790, 1995-1998 (H. Chen, "Concept-based Categorization and Search on Internet: A Machine Learning, Parallel Computing Approach"),
- NSF CISE Research Initiation Award, IRI-9211418, 1992-1994 (H. Chen, "Building a Concept Space for an Electronic Community System"),
- NSF CISE Special Initiative on Coordination Theory and Collaboration Technology, IRI-9015407,

1990-1993 (B. Schatz, "Building a National Collaboratory Testbed"),

- AT&T Foundation Special Purpose Grants in Science and Engineering, 1994-1995 (H. Chen), and
- National Center for Supercomputing Applications (NCSA), High-performance Computing Resources Grants, 1994-1996 (H.Chen).

We would also like to thank the group members of the Artificial Intelligence Laboratory and the undergraduate students in the Department of Management and Information Systems at the University of Arizona for their participation in the user evaluation. Their effort and time are greatly appreciated.

References

[1] T. Berners-Lee, R. Cailliau, A. Luotonen, H. F. Nielsen, and A. Secret. The World-Wide Web. *Communications of the ACM*, 37(8):76–82, August 1994.

[2] C. M. Bowman, P. B. Danzig, U. Manber, and F. Schwartz. Scalable internet resource discovery: research problems and approaches. *Communications of the ACM*, 37(8):98–107, August 1994.

[3] E. Carmel, S. Crawford, and H. Chen. Browsing in hypertext: A cognitive study. *IEEE Transactions on Systems, Man and Cybernetics*, 22(5):865–884, September/October 1992.

[4] H. Chen. Collaborative system: solving the vocabulary problem. *IEEE Computer*, 27(5):58–66, May 1994.

[5] H. Chen, P. Buntin, L. She, S. Sutjahjo, C. Sommer, and D. Neely. Expert prediction, symbolic learning, and nerual networks: An experiment on greyhound racing. *IEEE Expert*, 9(6):21–27, December 1994.

[6] P. DeBra and R. Post. Information retrieval in the World-Wide Web: making client-based searching feasible. In *Proceedings of the First International World Wide Web Conference '94*, Geneva, Switzerland, 1994.

[7] O. Etzioni and D. Weld. A softbot-based interface to the Internet. *Communications of the ACM*, 37(7):72–79, July 1994.

[8] G. W. Furnas, T. K. Landauer, L. M. Gomez, and S. T. Dumais. The vocabulary problem in human-system communication. *Communications of the ACM*, 30(11):964–971, November 1987.

[9] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, MA, 1989.

[10] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. The MIT Press, Cambridge, MA, 1992.

[11] P. Maes. Agents that reduce work and information overload. *Communications of the ACM*, 37(7):30–40, July 1994.

[12] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, Berlin Heidelberg, 1992.

[13] J. Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley Publishing Company, Reading, MA, 1984.

[14] B. Pinkerton. Finding what people want: experiences with the WebCrawler. In *Proceedings of the Second International World Wide Web Conference '94*, Chicago, IL, October 17-20, 1994.

[15] D. Riecken. Intelligent agents. *Communications of the ACM*, 37(7):18–21, July 1994.

[16] B. R. Schatz, A. Bishop, W. Mischo, and J. Hardin. Digital library infrastructure for a university engineering community. In *Proceedings of Digital Libraries '94*, pages 21–24, June 1994.

[17] B. R. Schatz and J. B. Hardin. NSCA Mosaic and the World Wide Web: global hypermedia protocols for the internet. *Science*, 265:895–901, 12 August 1994.

[18] M. M. Waldrop. Software agents prepare to sift the riches of cyberspace. *Science*, 265:882–883, 12 August 1994.