Applying an Evolutionary Algorithm to Web Search: the Methodology of *Evagent*

Wei Li
Faculty of Informatics and Communication
Central Queensland University
North Rockhampton, QLD 4702, Australia

Xinghuo Yu
School of Electrical and Information Engineering
RMIT University
Melbourne, VIC 3001, Australia

Abstract —An evolutionary algorithm is introduced to find authoritative resources on the Web. The problem of Web search is considered as an optimisation problem within hyperlinked space. We aim to find information that is both relevant and recent so as to cope with the dynamic nature of the Web. Theoretical studies have been made on problem-specific search space, fitness functions and genetic operators. The search space is constructed in the direction to the optimum, driven by the reproduction operator with good hubs as a clue. Fitness functions combine text-based and link-based analysis. The $(\mu+\lambda)$ evolution strategy just implements the selection scheme of elitism. The mutation operator helps to prevent search from trapping in local optimisation by introducing multiple domains. Experiments have been performed to study algorithm's performance. The algorithm has been implemented as a kernel component of an intelligent Web agent Evagent.

Keywords: Evolutionary algorithm, genetic operators, Web search agent, authority and hub.

1 Introduction

Searching on the Web for authoritative resources is the process of identifying Web pages that are highly relevant to given topics. We are unsatisfied with matches from commercial search engines. Inherent limitations of database size and updating rate prevent search engines from desirable coverage and recency. Intelligent search agents have emerged as a tool for addressing these problems. Although lots of theoretical studies have been made on intelligent Web search, evolutionary approach has rare been used. To our knowledge, *Webnaut* [6] utilizes a genetic algorithm to combine keywords and logic operators for query generation, and *MySpiders* [8] utilizes an evolutionary algorithm for managing a population of adaptive and autonomous Web crawlers, but they are in totally different meaning from the evolutionary methodology of this paper. When we define the Web search as an optimisation problem, an evolutionary algorithm is suitable to search for a solution. When using evolutionary approach to Web search, we have deeply investigated problem-specific evolutionary elements, and how to use them to form an effective logic for Web search. Experiments have been performed for suggesting proper parameters and demonstrating performance.

This paper is organized as follows. In section 2, we introduce related works and investigate their limitations as a context to develop our evolutionary algorithm for Web search. Section 3 describes the shape of topic-specific Web graph and its implications for evolutionary search on the Web. In Section 4, we deeply investigate evolutionary elements that contribute our evolutionary algorithm for Web search. In Section 5, discussions on the algorithm's performance has been made. Section 6 concludes the paper and gives our future research directions.

2 Web Search and Limitations

Basically, there are two Web search technologies, the text-based and the linked-based analysis. The text-based search usually uses the idea that the more often the term appears in a document; the more characteristic the term is for the document. The text-based search doesn't work well when Web pages are not self-descriptive or over self-descriptive. The creation of a hyperlink on the Web represents a concrete indication of the judgement: the creator of page $\mathcal{P}$, by including a hyperlink to page Q, has in some measure conferred authority on Q. Furthermore, a topic can be roughly divided into pages with good content on the topic, called authorities, and directory-like pages with many hyperlinks to pages on the topic, called hubs. Recursively, a document that points to many good authorities is an even better hub, and a document pointed to by many good hubs is an even better authority. The idea, "what do the neighbours think about it and how many neighbours prefer it?", comes from the fact that it uses the content of other pages to rank the current page.

This research is partly supported by project *Distributed Computation Based on Agents*, the Ministry of Education of China.

Search algorithms [1, 7, 9] based on hyperlink analysis has been created and integrated into some search engines (Google, TOPIC, WTMS). These search engines do work well than only text-based search engines. Here again we ask: are you satisfied with them? The answer would be no or no yet. It is hard for a search engine to keep up with the Web growth and change rates. Intelligent agents have emerged as a tool for addressing these problems. Intelligent Web agents are running on client side, and they use query-dependent ranking algorithms. The basic idea is to construct a query-specific graph, called neighbour graph, and perform hyperlink analysis on it. Ideally this graph will contain more topic-relevant pages. In fact, crawling the whole Web for constructing a topic-specific neighbour graph is not impractical. Intelligent agents exploit a promising neighbour graph as a snapshot of the topic-specific Web for searching the strongest authorities. [4] exploits the following procedure to do that.

1) Collect the n (usually 200) highest-ranked pages for a given query from a text-based search engine such as AltaVista. These n pages are referred as the *root* set.
2) Expand the *root* set into the *base* set by including any page pointed to by a page in the *root* set and any page that points to a page in the *root* set.
3) Each document is modelled by a node. There is a directed edge from page P to page Q if page P hyperlinks to page Q.

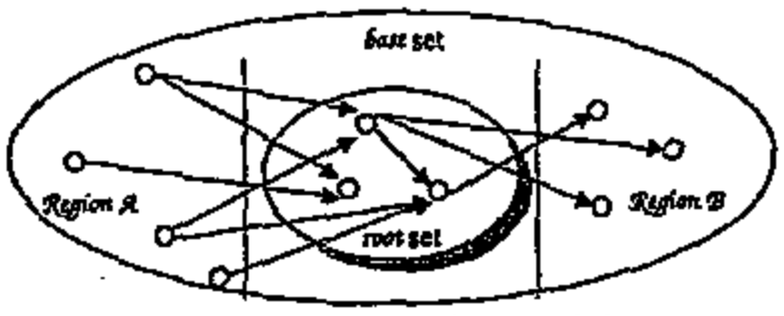

Fig. 1. Constructing a promising neighbour graph

Typically, the *base* set has Web pages in the range of 1,000~5,000. [4] thought that the strongest authorities would hopefully occur in the *base* set. [1] exploits the same method as that of [4] but performs the expansion step twice. And the expanded set has about 3,000 distinct pages, depending on the topic. [5, 9] use similar methods as that of [4] with the expanded sets having about 1,000 distinct pages. In fact, the algorithm of [4] doesn't claim to find authoritative enough pages for a query. There are three problems with the algorithm, which could be the potential reasons for generating unsatisfied matches.

1) The base set is not large enough for identifying the strongest authorities.
2) Pages that share the same children with the pages in the root set, or pages that share the same parents with the pages in the root set, are not included in the neighbour graph.
3) The neighbour graph is incomplete for identifying the strongest authorities within it.

How promising is the neighbour graph? The fist limitation is evident when comparing the neighbour graph with the whole Web. For the second limitation, [2] suggests that pages with the same parents or with the same children, called co-cited pages, might be topic-relevant. Those pages would be promising candidates of the strongest authorities. The third limitation is not evident. In order to compute a page's authority score precisely, we need its all parent hyperlinks. And in order to compute a page's hub score, we need its all children hyperlinks. When the algorithms of [4, 5] are applied to the above neighbour graph, the authority scores of pages in *Region A* will be zero because of their lack of parents. And the hub scores of pages in *Region B* will be zero because of their lack of children. Further, the computation for hub scores of Web pages in *Region A* is not precise, because they only have children in the *root* set, which are not their total children. Neither is the computation for authority scores of Web pages in *Region B* because they only have parents in the *root* set, which are not their total parents. After the computation, the strongest authorities could occur only in the *root* set or *Region B*, and the strongest hubs could occur only in the *root* set or *Region A*. But the case is not the same as in the real Web, where any page is not a great grandparent without any ancestor and any page is not a great grandson without any descendant. The neighbour graph is incomplete to be a snapshot of the hyperlinked Web.

3 Topic-Specific Hyperspace

How do we construct a topic-specific Web graph so that our algorithm could be both quite effective and precise for hub and authority scores? There is no reason for all good hubs to be adjacent

to the *root* set. Different hubs are more likely to be found among the "sibling" of parents and even grandparents of the *root* set. Excluding the grandparents of the *root* set may possibly leave a number of potentially good hubs. In the absence of good hubs, it would be harder to find pages that have good authorities.

Fig. 2 shows an example Web graph from [3]. Without loss of generality, we can suppose that it is a topic-specific Web graph. For a search agent, it is impractical to afford downloading the entirely topic-specific Web. We exploit different "tricks" to find a set of starting points from which the entirely or the most parts of topic-specific Web can be reached after certain iterations.

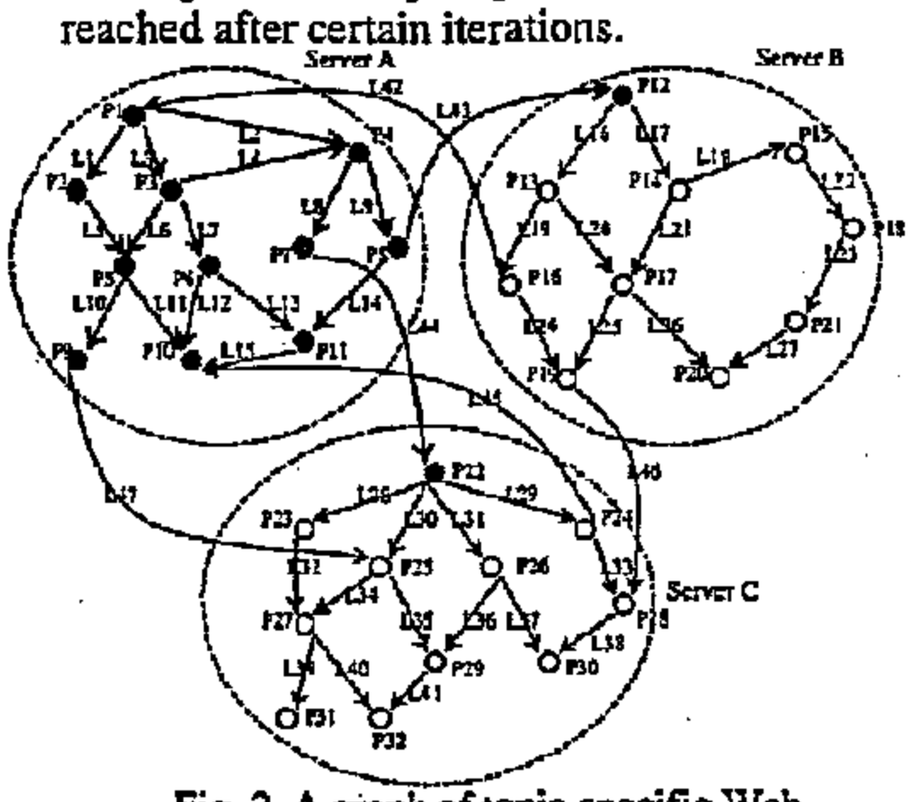

Fig. 2. A graph of topic-specific Web

We define a strategy, called *Fixed-Depth Search* (*FDS*) for the Web graph traversal. From a given vertex v, the *FDS* strategy proceeds along all possibly directed paths from v as deeply into the graph as possible within a given depth d. An example is that if the start vertex is $P1$ and the depth is 3, the subgraph covered by *FDS* is illustrated in Fig. 2 with bold vertexes and edges. If we compute the hub and authority scores for the covered pages, we find that the hub scores of pages $P9$, $P11$, $P12$, $P22$ are not precise, and the authority scores of pages $P1$, $P10$ are not precise. Those cases are due to that we couldn't cover their all outgoing or ingoing links. However, if we continue our *FDS* process for multiple generations with each generation from different start pages, the cases are quite different. From Table 1, we found that we just iterate *FDS* three generations, however we get 100% page coverage, 97.9% link coverage, and 96.9% pages' hub and authority scores are computed precisely.

Table 1. Iterating FDS for good coverage

	G1	G2	G3
Start page(s)	$P1$	$P5$-$P6$	$P13$, $P14$, $P23$-$P26$
Imprecise hubs	$P9$, $P11$, $P12$, $P22$	$P15$-$P17$, $P27$-$P29$	$P21$
Imprecise authorities	$P1$, $P10$	$P1$, $P28$, $P30$	$P20$
Covered pages	40.6%	81.3%	100%
Covered links	34.0%	74.5%	97.9%
Precise hubs	69.2%	76.9%	96.9%
Precise authorities	84.6%	88.5%	96.9%

With this methodology, we can overcome the investigated limitations because:

1) in each generation, we explore almost the same size space as those of [4, 5]. And after certain generations, we hope to explore a large enough space for identifying the strongest authorities.
2) at a certain generation, the Web pages in *Region A* and *Region B* could move into the *root* set at next generation because we change the seed URLs for next generation.
3) if Web pages in *Region A* and *Region B* move into the *root* set at next generation, they would get more precise computation of their authority and hub scores then.

4 Evolutionary Algorithm for Web Search

We use an evolutionary algorithm to implement our Web search idea. Firstly we categorize the Web search tasks into elements of the algorithm, which are summarized in Table 2.

Table 2. Web search tasks and their corresponding evolutionary elements.

Search tasks	Evolutionary elements
Selection of start pages	Individual selection
FDS traversal	Reproduction
Hub and authority computation	Fitness functions
Identify the most authoritative	The $(\mu+\lambda)$ evolution strategy
Web domain diversification	Mutation

4.1 Fitness Functions

Fitness values for a Web page are its *Hub* and *Authority* scores. The hub and authority scores are the de facto criteria for judging a Web page's reputation. Our algorithm for the fitness values is as follows. It uses the same strategy as the algorithm of [9]. Let's view any collection V of hyperlinked pages as a directed graph $G=(V, E)$: the nodes correspond to the pages, and a directed edge $\langle P, Q\rangle \in E$ indicates the presence of a hyperlink from page P to Q. The t is a term in a

query. Then the authority score $A(P, t)$ and the hub score $H(P, t)$ of a page P are computed by:

```
for ( each <Q, P>∈E) A (P, t)=ΣH(Q, t)/O(Q))
for (each <P, Q>∈E) H(P, t)=ΣA(Q, t) /I (Q))
```

Where $I(Q)$ is the number of topic-relevant ingoing hyperlinks to Q, and $O(Q)$ is the topic-relevant outgoing hyperlinks from Q. The two functions above quantify the hubs and authorities and their relationship, by which hubs and authorities reinforce one another.

Many Web pages are self-descriptive. That means the topic words of a page probably occur in its HTML textual contents, especially in the *Title*, or the *Meta* tags, such as *Keywords*, *Description* or *Distribution*. Here we define two vectors *Field* and *Score*. An element of *Field* represents a token; and an element of *Score* represents a score assigned to the corresponding token's content.

Table 3. Definitions of HTML structured fields and their self-description scores.

1	1	2	3	4	5
Field	Title	Keywords	Description	Abstract	Pagetext
Score	10	10	10	10	10
Freq.	1	1	1	1	15

So the self-description score $sd(P, t)$ of a Web page could be computed by the following procedure.

```
sd(P, t)=0;
for ( i∈[1, 2,..., 5])
    if (t occurs in Field(i) k times)
sd(P, t)+=Min(10, ( k* Score(i)/ Freq(i)))
```

It is believed that the text around a *href* link to a page P is good description of the contents of P by other pages [1]. So we open a window around the tag ..... with 50 bytes wide, and compute the hyperlinked-description score $hd(P, t)$ of a Web page P.

```
hrefScore=0;
for each topic-relevant hyperlink L to P
   if ( the window around L contains term t)
        hrefScore+=5;
hd(P, t)=Min(50, hrefScore)
```

4.2 Population Initialisation

The first stage of the evolutionary is the population initialisation. The initial population just consists of seed URLs and terms provided by a user or *Evagent* itself. If a user does not provide enough seed URLs, *Evagent* can make it up to the required number. Good seed URLs are highly topic-relevant hubs and are usually from different domains. What we should keep in mind is that it is appropriate to start with several hubs that are selected heuristically. The evolutionary algorithm is thereby initially aimed in promising directions. In our evolutionary algorithm, we symbolize the initial population as $PL=(SU, T)$, where SU denotes the set of the provided seed URLs, and T denotes the set of the provided terms.

4.3 Reproduction

The second stage is the reproduction of PL. The first step of the reproduction, which we called the expansion, is to construct a promising subgraph $G=(V, E)$ of the Web. This expansion could be finished by crawling Web pages in SU and expanding SU along all hyperlinks that leave it to a certain depth d, i.e., the procedure of applying *FDS* strategy for one iteration.

```
for (each page P∈SU)  {P.depth=0; P.visited=false;};
V=SU; E=Φ;
For ( each P ∈V and P.depth<=d and ! P.visited)
   { crawl(P); P.visited=true;
     for (each topic-relevant hyperlink <P, Q> in page P)
          {add <P, Q> into E; Q.depth= P.depth+1;
            Q.visited=false; add Q into V}}
return G=(V, E);
```

The second step of the reproduction is to compute the fitness values for each page in $G=(V, E)$ by the following procedure.

```
for ( each page P∈V and t∈T)
 {i=0;
  while i<k
  {for (each <Q, P>∈E) A(P, t)=A(P, t)+H(Q, t)/O(Q);
   for (each <P,Q>∈E)  H(P, t)=H(P, t)+A(Q, t)/I(Q);
   i++; }}
The authority score of P is A(P, t)+sd(P, t)+hd(P, t);
The hub score of P is H(P, t);
```

The third step of the reproduction is to form a new population NPL. In order to do that, we filter out $G=(V, E)$ into four sets of pages, which are *top50Hub*, *top10Authority*, *hubCover* and *topHub*. The definitions of them are as follows. Where $|X|$ denotes the cardinal number of the set X. So our new population can be define as $NPL=\{top10Authority, topHub\}$.

```
Top50Hub={P| P ∈V and H(P, t) is one of the top 50 hub scores}
top10Authority={ P| P ∈V and A(P, t)+sd(P, t)+hd(P, t) is one of
                  the top 10 authority scores}
cover(X)={Q| P∈X and X⊆V and X≠Φ and <P, Q>∈E}
CS={X| X⊆V and X≠Φ and cover(X)= V}
for (each X∈CS)  mcn=Min(|X|)
If (X∈CS and |X|=mcn) hubCover=X
topHub=top50Hub∧hubCover
```

4.4 Individual Selection

At this stage, what we are going to do is to select pages with good fitness values for different purposes. Reasonably, pages with higher hub

scores are good candidate for seed URLs; and pages with higher authority scores are good candidates for the final matches. Normally, we directly select the pages of *top10Authority* into a set $\mathcal{AC}$ as authoritative candidates, i.e., we prefer the $(\mu+\lambda)$ evolution strategy for authority selection. Pages of *topHub* can be divided into $d+1$ (from 0 to d) levels corresponding to the depth for *FDS* traversal. Pages with low levels are not suitable to be seed URLs for next generation because the next Web subgraph originated from them by *FDS* will overlap the last subgraph in most parts. Here we define the following procedure to control the selection of new seed URLs, which comprise the set $\mathcal{NS}$.

```
offset=( d+1)/2; loop=0;
while( loop<=1) {
 for (each page P in topHub with hub scores
       in descending order)
 if (P.depth>=offset and P is not in NS and |NS|<sn )
      select P into NS;
 if (|NS|<sn) offset--;
loop++;}
return NS;
```

Where sign *sn* is the required number of seed URLs. The above selection process represents our following ideas. New seed URLs should have a certain offset in depth from the old ones. That guarantees that we could explore different hyperspace in next generation.

4.5 Mutation

The final stage of the algorithm is the mutation. In the context of Web search, we exploit the mutation to increases the diversity of domains for searching more different parts of the Web. We exploit variable mutation probability for different generations so that we can keep the number of different domains of seed URLs being constant. To do the mutation, we maintain a list $\mathcal{D}$ of the domain names visited so far since the first generation. *Evagent* queries the famous commercial search engines, for example Yahoo, AltaVista or Google etc., with term t as keywords. We think the top-ranked pages returned by them could be promising ones as the mutation seed URLs. With variable probability, the pages in $\mathcal{NS}$, which have low hub values and have domain names in $\mathcal{D}$, will be replaced by the mutation URLs. In the following procedure, we use sign $\mathcal{MS}$ to denote the set of mutation URLs, sign *dn* to denote the required number of different domains, and sign *sn* to denote the required number of seed URLs. Going through the entire evolutionary procedure one generation is said to produce a new generation $\mathcal{NG}=\{\mathcal{NS}, top10Authority\}$.

```
while (|domain(NS)-D|<dn) {
 if (P∈MS and domain(P)∉(domain(NS)∪D)  add P into NS; )
while (|NS|<sn) {
 if (P∈MS and domain(P)∉(domain(NS)∪D)  add P into NS; )
while (|NS|>sn) {
if (domain(P)∈domain(NS)∩D and P is with the lowest hub
     values)  delete P from NS;}
D=D∪domain(NS);
```

4.6 The Number of Generation

It is often a trial-and-error process to select the number of the generations that enables evolutionary algorithm converge. In our evolutionary algorithm, from the user's initial input $\mathcal{PL}=(\mathcal{SU}, \mathcal{T})$, we define the following procedure to control the number of generations.

```
LS=SU;LA=Φ;NA=Φ; AC=Φ; g=0;
while (g<4 and the number of generations<maxNum)
  { if (the number of shared URLs with the same
       fitness between LA and NA is more than 80%) g++;
  else g=0;
  LA=NA; Compute a new generation NG from LS;
  AC=AC∪ top10Authority; NA= top10Authority;
  LS=NS;}
return top 10 of AC as final matches;
```

The iteration will be ended whenever the case, which the new *top10Authority* and the last *top10Authority* share at least 90% individuals with the same fitness, persists 4 generations, or max generation number comes first. The top 10 authorities of final $\mathcal{AC}$ are returned matches to user's queries.

5 Searching with Evagent

We have conducted a lot of search examples for testing the algorithm's performance. In this section, we report a typical search example. The search example has been performed by several groups of parameters for different evaluations. We use 3 functions to compute numerical values for making convergence or performance curves.

1) $F(P, g, t)=A(P, g, t)+sd(P, g, t)+hd(P, g, t)$
2) $SN(g, g+1)=\Sigma\ i$, where if $P \in top10Authority(g) \cap top10Authority(g+1)$, i is assigned to1; otherwise i is assigned to 0.
3) $DN(g, g+1)=\Sigma\ j$, where if $P \in top10Authority(g) \cap top10Authority(g+1)$ and $F(P, g, t)\neq F(P, g+1, t)$, j is assigned to 1; otherwise j is assigned to 0.

Given a Web page P and a search topic t, the first function can compute the page's authority score for generation g. The second function can compute the number of shared top authorities between vicinity generation g and $g+1$. In fact,

the second function can examine the convergence speed of the algorithm. The third function can compute the number of shared top authorities with different authority scores between vicinity generation g and $g+1$. Actually, the third function can test the convergence saturation of the algorithm.

The algorithm was tested by 3 groups of parameters on search topic: *Mobile Agents*, which are autonomous, intelligent programs that move through a network, searching for and interacting with services on the user's behalf.

Table 4. Parameters for performance test

Parameters	Group1	Group 2	Group 3
Number of seed URLs	20	5, 10, 25	20
Number of different domains of seed URLs	15	5	10, 15, 20
Number of generations	20	20	20
Traversal depth	5, 7, 9	9	9

5.1 Effectiveness

In this experiment, we use parameters of Group 1 to test the effectiveness of the algorithm. Within 20 generations, *Evagent* tested 54,396 hyperlinks within 300 different domains and identified 705 topic-relevant pages. The evolutionary algorithm has good effectiveness. We make this conclusion from numerical results, which can be explained from the following two angles. Firstly, the final top 10 authorities are highly topic-relevant.

Table 5. Top 10 authorities for *Mobile Agents*

http://www.research.ibm.com/massive/ *Mobile Agents: Are they a good idea?, IBM Research Report*
http://www.omg.org/cgi-bin/doc?orbos/97-10-05 *Mobile Agent System Interoperability Facilities Specification by GMD FOKUS*
http://sirio.dsi.unimo.it/MOON/papers/papers.html#Paper5 *Mobile Agent Technology: Current Trends and Perspectives, a research paper by G. Cabri*
http://www.cs.dartmouth.edu/~dfk/papers/kotz:future2/ *Mobile Agents and the Future of the Internet by David Kotz and Robert S. Gray*
http://www.trl.ibm.com/aglets/index_e.htm *Aglets, a mobile agent system by IBM*
http://agent.cs.dartmouth.edu/ *D'Agents, a mobile agent system by Dartmouth College*
http://www.recursionsw.com/products/voyager/voyager.asp *Voyager, A Mobile agent System by Recursion Software, Inc.*
http://www.toshiba.co.jp/plangent/index.htm *Plangent, a mobile system by Toshiba*
http://www.cetus-links.org/oo_mobile_agents.html *The Cetus Links on mobile agents.*
http://mole.informatik.uni-stuttgart.de/mal/mal.html Mole, a mobile agent list by University of Stuttgart, Germany

We have the URLs checked by researchers on mobile agents at our Faculty. They are satisfied with both the authority of result URLs and their ranks. Secondly, fitness values increasingly synchronize the number of generations. Convergence curves are illustrated in Fig. 3. Although the fitness values are quite different for different depths and vary severely at first generations, they vary slightly and converge progressively to a constant after certain generations.

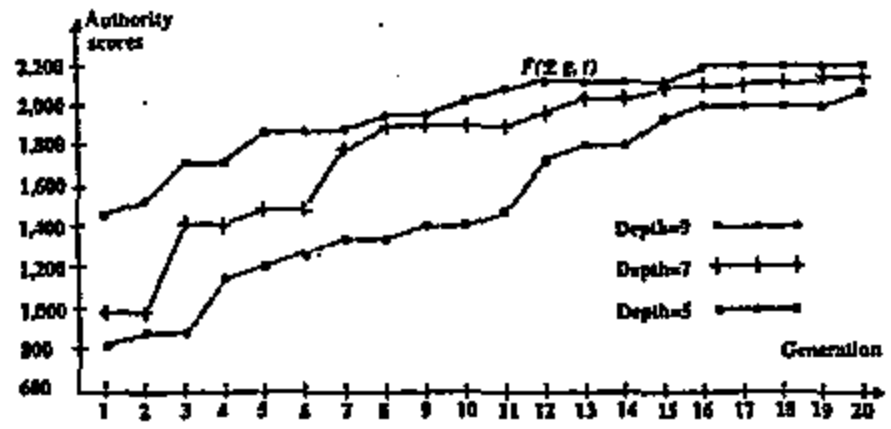

Fig. 3. Fitness curves from Group 1

5.2 Traversal Depth

In this experiment, we also use parameters of Group 1 to examine how the traversal depth contributes to the convergence of our evolutionary algorithm.

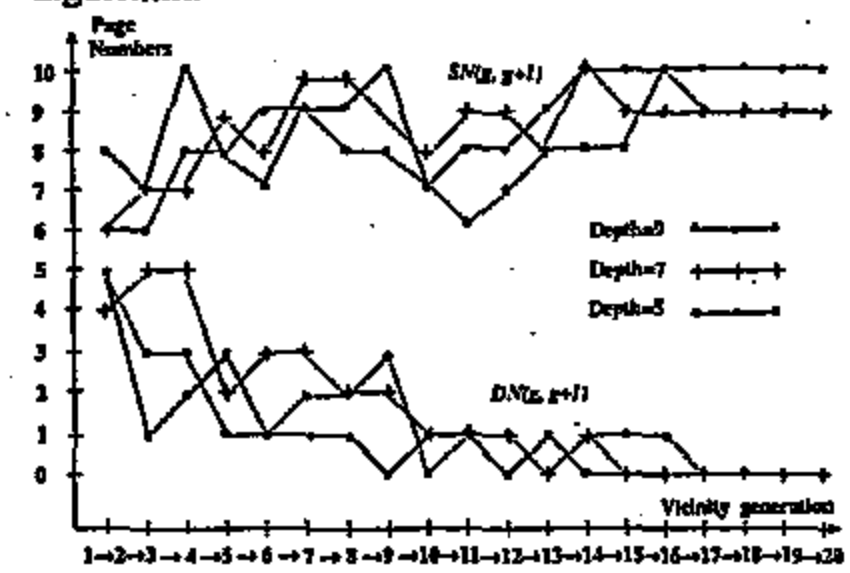

Fig. 4. Convergence curves for different depths

Fig. 4 shows that large depth makes a significant contribution to the algorithm's performance. If the depth is small, the convergence curve fluctuates for more generation before it converges to a steady status. With large depth, quick convergence followed by saturation takes place.

5.3 Other Evolutionary Elements

Size of initial population for each generation is the number of seed URLs in our evolutionary algorithm. In order to determine how the number of seed URLs contributes to convergence, we use parameters of Group 2, which keeps other parameters constants except for the number of seed URLs. Convergence curves (omitted because of the length of the paper) show that large number of seed URLs makes a significant contribution to our evolutionary algorithm's performance. Within

20 generations, Only the computation with 25 seed URLs gets good convergence. Small number of seed URLs less than 10 fails to converge with satisfied fitness.

The influence of the mutation probability on our evolutionary algorithm's performance is also examined. In order to determine how the mutation probability contributes to convergence, we use parameters of Group 3. We make the mutation probability varying from generation to generation so that numbers of different domains of seed URLs are kept as constants 10, 15 and 20 respectively for 3 computations. From the convergence curves (omitted because of the length of the paper), we conclude that large mutation probability, i.e., large number of different domains of seed URLs contributes significantly to the algorithm's performance.

6 Conclusion and Future Work

We couldn't just rely on the textual contents of a Web page to figure out what topic it is about, because many Web pages are not self-descriptive. However, link-based search engines suffer from other limitations: their promising subgraph of the Web is not only not large enough but also incomplete for computing authority score for identifying strongest authorities within it. Our evolutionary approach can overcome text-based and link-based search engines' limitations by:

1) mining the descriptive information about a Web page not only by its textual contents but also by link texts using our fitness functions;
2) building a promising space by changing seed URLs and reconstructing the subgraph of the Web by our reproduction operation;
3) increasing the precision of the authority and hub scores of a Web page by introducing more potential parents and children of the page when a new subgraph is reconstructed.

The experimental outcomes suggest that the evolutionary methodology is really a good idea for the Web search. Analysis to the numerical results bring us to the following conclusion. Given a topic, the size of the topic-specific Web is almost constant during a very short period. Because of this reason, evolutionary elements, such as large traversal depth, large size of initial population, and large mutation probability, contribute significantly to the convergence and performance of our evolutionary algorithm. The underlying principle, which makes *Evagent* highly effective, is that with evolutionary paradigm, we are not just for optimising; we are also creating conditions in which optimisation occurs.

Future research needs to focus on a deeper level of understanding the effectiveness and performance of our evolutionary algorithm and exploiting this information for the methodology improvement. Another area of future research can focus on contributing *Evagent* to more Web applications, such as Search-by-Example, Mirrored Hosts Finding, and Web page Categorization. Future work will also focus on moving *Evagent* from client side as an agent to server side as a search engine. More search agents like *Evagent* are distributed, and they can search different parts of the Web in parallel. By this parallel computation, we can cope with the dynamic nature of the Web very well.

References

[1] Soumen Chakrabarti, Automatic resource compilation by analysing hyperlink structure and associated text, In Proceedings of 7th International WWW Conference, Brisbane, Australia, 1998.

[2] Jeffrey Dean, Monika Henzinger, Finding Related Pages in the World Wide Web, In Proceedings of 8th International WWW Conference, Toronto, 1999.

[3] Paul Debra, Finding Information on the Web, CWI Quarterly, Vol. 8, nr. 4, pp. 289-306, 1996

[4] Jon M. Kleinberg, Authoritative Sources in a Hyperlinked Environment, In Proceedings of 9th ACM-SIAM Symposium on Discrete Algorithms, 1998.

[5] Alberto O. Mendelzon, Davood Rafiei, What do the Neighbour Think? Computing Web Page Reputations, IEEE Data Engineering Bulletin, September 2000.

[6] Zacharis Z. Nick, Panayiotopoulos Themis, Web Search Using a Genetic Algorithm, IEEE Internet Computing, Vol. 5, Issue 2, April 2001

[7] Lawrence Page, The PageRank Citation Ranking: Bringing Order to the Web, http://www.stanford/~backrub/pageranksub.ps.

[8] Gautam Pant, Filippo Menczer, MySpiders : Evolve your own intelligent Web crawlers, to appear in Autonomous Agents and Multi-Agent Systems Journal, available at http://dollar.biz.uiowa.edu/~fil/papers.html, 2002

[9] Davood Rafiei, Alberto O. Mendelzon, What is this Page Know for? Computing Web Page Reputations, In Proceedings of 9th International WWW Conference, Amsterdam, May 2000.