Evolving Strategies for Focused Web Crawling

Judy Johnson JAJ@NEC-LABS.COM
Kostas Tsioutsiouliklis KT@NEC-LABS.COM
NEC Laboratories America, Inc., 4 Independence Way, Princeton, NJ 08540 USA

C. Lee Giles GILES@IST.PSU.EDU
School of Information Sciences and Technology, The Pennsylvania State University, 001 Thomas Building, University Park, PA, 16802, USA

Abstract

The rapid growth of the World Wide Web has created many challenges for both general purpose crawling, search engines and web directories, making it difficult to find, index, and classify web pages based on a topic. Topic driven crawlers can complement search engines because they pre-classify the pages retrieved by the crawl. To implement such a focused crawler, a strategy for ordering the crawl frontier is required. Such a strategy can only use information gleaned from previously crawled pages to estimate the relevance of a newly observed URL. Because the best strategy for ranking URLs in the crawl frontier is not immediately apparent, we discover strategies by evolving them using a genetic algorithm. Strategies are learned by evaluating the results of crawls simulated using a database generated by a previous, more general crawl. We conclude that a rank function that combines analysis of text and link structure yields effective strategies. The evolved strategies perform better than the commonly used Best First strategy.

1. Introduction

The World Wide Web is experiencing an exponential growth both in number of users and in size. Google currently indexes approximately $3,000,000,000$ web pages (google, 2003). Web directories such as Yahoo or the Open Directory Project (dmoz) are unable to categorize more than a fraction of available pages due to the need for human classification. A focused crawler that dynamically browses the web looking for pages related to a particular topic can eliminate some of the problems created by the size of the web as a whole. Pages retrieved are pre-classified and are devoted to one topic. Thus focused crawling is ideally suited to generate data in response to the needs of an individual user or a community of users interested in one topic.

Topic driven crawlers rely on a strategy for choosing which URLs to download during the course of a crawl. Because of the size of the web, discovering good strategies for finding topically relevant pages is difficult. The crawler may have to traverse irrelevant pages to reach highly relevant pages. Because the crawler only has information about the pages it has already crawled, predicting which pages lead to good pages is not an easy task. A good strategy needs to be able to determine when to pursue links even though the source page may not be very relevant. For these reasons learning a strategies for focused crawling may find better strategies than attempting to use strategies that seem promising. Because of bandwidth limitations and the need to limit the number of times a given page is downloaded, it is difficult to test many strategies on a large scale. Previous work has focused on evaluating a few strategies that seem to have promising characteristics.

Early work in focused crawling includes (Chakrabarti et al., 1999; Diligenti et al., 2000; Menczer et al., 2001) which evaluates several crawl strategies, finding that a naive, Best First strategy performs the best. More recent work on focused crawling includes (Chakrabarti et al., 2002; Menczer et al., 2003; Pant & Menczer, 2002; Srinivasan et al., 2003). Our approach is to evolve a strategy based on the text and link characteristics of the referring pages. The strategies we evolve produce a rank function which is a weighted sum of the text and link scores.

Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003), Washington DC, 2003.

We use a genetic algorithm to evolve the weights for this function. Genetic algorithms are randomized, parallel search algorithms that search from a population of points (Goldberg, 1989; Holland, 1975). Individuals in the population are evaluated for fitness, then propagated to later generations by means of probabilistic selection, crossover and mutation. In this instance the individuals in the population are the strategies to be tested. The population evolves to find better strategies by selecting the most fit strategies, or the strategies that retrieve the most related pages over the course of a simulated crawl, and propagating these strategies to the next generation. Because of the random search properties of genetic algorithms, they provide an efficient tool for solving problems with large, poorly understood search spaces (Goldberg, 1989; Holland, 1975). Genetic algorithms dynamically balance exploration of the search space with exploitation of promising areas in the space through recombination operators, crossover and mutation. Much of the research in the area of focused crawling has concentrated on testing a hypothesis strategy on line. Because repeatedly crawling the same web sites is not considered good etiquette, it is difficult to test a large number of strategies on pages related to one topic. We combine the genetic algorithm approach, which allows us to explore many possible combinations of potential indicators of good pages, with simulations on a large database of previously downloaded pages. The genetic algorithm allows us to explore the space of potential strategies, without preconceived notions of what the best indicators of topically related pages may be. As the run progresses and good strategies are discovered, the genetic algorithm concentrates on strategies that give heaviest weight to the most promising features.

The rest of the paper is organized as follows. A description of the crawl simulation and evaluation which describes the crawl databases, classification methods and fitness function is in Section 2.1. The rank function used by the strategies is described in Section 2.2 and results for simulations run on two different topics are presented in Section 3. Section 4 presents conclusions and directions for future work.

2. Crawl Simulation and Evaluation

2.1. The Simulation

The simulator is shown in Figure 1. There are two inputs to the system. First is a database that is built from the data collected by previous breadth-first search crawls. A seed set is also input to the simulator. The seed set is a list of URLs that are known to be relevant to the topic. The database is built from previ-

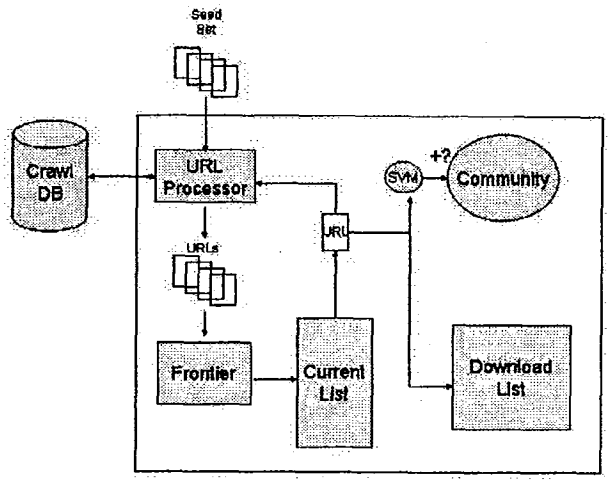

Figure 1. The simulator

ous crawls that have been run starting from a seed set of URLs relating to a topic, then crawling out to several levels away from the original seeds. The database includes an index representing the graph of the link structure between the downloaded pages and also indexes that contain feature data about the text on the pages downloaded. Features are words or bi-grams that appear on the relevant pages. The database allows a simulation of a crawl to be run based on the web graph index and the features of the pages. Pages are classified using a support vector machine (SVM) trained on the features of a seed set of topically related URLs and a randomly chosen set of negative pages. SVMs have been shown to be good classifiers of text documents in (Drucker et al., 2001; Joachims, 1998).

The simulator also maintains several lists of URL data. First, it keeps a current list, which is the list of URLs that the simulator is currently downloading. It also maintains the crawl frontier list; URLs that have been seen during the course of the crawl but not yet added to the current list. The data associated with these URLs includes a rank score that determines when or if a given URL will be added to the current list. Once a URL has been selected to return from the current list, it is added to a downloaded list. This list is used to determine the success or fitness of the strategy used in a simulation. Pages that are marked by the SVM as positive, or related to the topic, are also put into the topic community list.

The simulated crawl begins by processing the URLs in the seed set then adding them to the current list. The maximum size of the current list is set to be 100

with new URLs added whenever the count of the remaining URLs decreases to 50. The current list size is kept small so that the current list is not populated by low ranked URLs when higher ranked URLs may have been seen later in the crawl. URLs are ranked according to a rank function determined by the strategy used. The 50 highest ranked URLs are added to the current list.

Most real crawlers asynchronously download a list of URLs. Several connections are open simultaneously. Which page returns first is based on a combination of network factors, traffic at the host, and also the amount of time the connection has been open. To emulate this behavior, URLs are randomly returned or "downloaded" in a probabilistic manner based on how long they have been in the current list. When a URL is added to the current list it is assigned a time stamp of one. Every time new URLs are added to the current list, the time stamps of the remaining URLs are incremented by one. A simple roulette wheel selection is used to choose the next downloaded URL; a URL that has a higher time stamp has a higher probability of being chosen to be "downloaded".

Once a URL has been chosen as the downloaded URL, it goes to the URL processor. The ids of its out links are retrieved from the web graph database. These new URLs are then processed and their ranks calculated. Processing includes retrieving feature data and calculating the SVM score for the page. Also, scoring factors such as the hubs, authorities, and community scores are updated before calculating the ranks. The new URLs are then added to the frontier to be available for adding to the current list. If the downloaded page is determined to be related to the topic by the trained SVM, it is added to the topic community list maintained by the simulator.

For each strategy, the simulation is run a number of times. The number of runs per strategy is a user parameter to the system. The fitness of a given simulation is

$$Fitness = \frac{GoodPageCount}{TotalPageCount} \quad (1)$$

The overall fitness of an individual or strategy is the average of the simulation fitnesses over several runs.

2.2. The Rank Function

The rank function is a weighted combination of several scoring functions.

$$Rank(u) = \sum_{i=1}^{n} w_i * S_i(u) \quad (2)$$

subject to $\sum_{i=1}^{k} w_i = 1$. Because u has not been seen by the crawler the scores are calculated based on what is known about the parent pages of u. The set of parent pages is defined by $P_u = \{v \in D : u \in U, (v,u) \in E\}$ where U is the set of URLs observed so far, E is the set of edges or links between URLs in the URL set, and D is the set of URLs already downloaded. Scores can depend on the text of the pages in P_u or the link structure including the set of out links of $p \in P_u$, $L_p = \{v \in U : (p,v) \in E\}$

The first score, S_1, is a simple estimation of a hub score. Hubs have been defined to be pages that point to a large number of authority pages for a given topic (Kleinberg, 1999). We estimate a hub by the number of out links on the parent page divided by the average number of out links on all of the pages already downloaded by the simulation.

$$HUB = S_1(u) = \frac{|L_p|}{AveOutCount}, P \in P_u \quad (3)$$

where

$$AveOutCount = \frac{\sum_{i=1}^{N} |L_i|}{N}, i \in D, N = |D| \quad (4)$$

In our simple score, a high hub score indicates a page that is pointed to by a page with a higher than average number of out links. The URLs in L_p get a higher score if $|L_p|$ is higher than average.

Similarly, the second score is an estimate of an authorities score. Authorities have been defined as pages that are linked to by many hubs (Kleinberg, 1999). We use a simple estimate of the number of pages linked to u divided by the average in count for all of the pages seen.

$$AUTHORITY = S_2(u) = \frac{|P_u|}{AveInCount} \quad (5)$$

where

$$AveInCount = \frac{\sum_{i=1}^{M} |P_i|}{M}, i \in U, M = |U| \quad (6)$$

In our simple estimate, a URL with a high authorities score is one that is pointed to by more than the average number of pages; $|P_u|$ is higher than average.

The next two scores attempt to capture the concept of a web community based on link structure. Link structure has been shown to be effective in finding clusters of topically related pages (Flake et al., 2000; Gibson et al., 1998). The community C is built as the simulation progresses by adding any positively identified pages that have been downloaded to the set. The original seed set is also included in the community. The

community in link score is calculated by

$$COMM_{in} = S_3(u) = \frac{|P_u \cap C|}{|P_u|} \quad (7)$$

The in link community score is the number of community pages that point to the current URL normalized by the total number of pages pointing to the current page. In this case we can use the in links to u because we know from the downloaded pages which ones links to u.

The community out link score is based on the URLs that the parent pages of the current URL link to. We use the source page out links because we don't know the out links on u.

$$COMM_{out} = \quad S_4(u) = \sum_{i=1}^{p} \frac{|L_i \cap C|}{|L_i|}, \quad i \in P_u, p = |P_u| \quad (8)$$

The out link score is the number of links pointing into the community normalized by the number of links on the parent pages. Whenever a new page is downloaded that points to a given URL, the two community scores are recalculated and the rank is updated.

The remaining scores are the SVM scores for the parent pages of the URL under consideration going back k generations, where k is a user parameter. The set of parent pages at the k^{th} level, P_u^k is defined by

$$P_u^1 = P_u \quad (9)$$

,

$$P_u^k = \{v : v \in P_j, j \in P_u^{k-1}\} \quad (10)$$

Since a URL may be pointed to by more than one page, we take

$$PARENT_k = S_{4+k} =$$

$$MAX(\{abs(SVM(p_i)), p_i \in P_u^k\}) \quad (11)$$

to be the parent score if the signs of the SVM score are the same. Thus if both parents are positive the maximum score is used, if both are negative the smallest score is used. If the signs differ, we take the maximum score.

$$PARENT_k = S_{4+k} =$$

$$MAX(\{SVM(p_i), p_i \in P_u^k\}) \quad (12)$$

3. Results

The system was tested using two databases. The first was built from the results of a crawl of 2.7M URLs on the subject of middle eastern dancing. This crawl was generated by starting with 13 middle eastern dance URLs then doing a breadth-first search crawl to five links from the seed set. The second database was built from a crawl of 3.5M URLs that was derived from a breadth first search crawl starting from 400 URLs on the topic of baseball to a depth of 4 links from the seed set. This set also included back links pointing to the original seed set and the set of URLs one link away. Of the two sets, the baseball set contains a much higher percentage of relevant pages because of the higher popularity of baseball and because the original crawl also included the back links.

A set of 139 URLs was used as a positive set for training both the SVM used for classifying the middle eastern dance pages and as a seed set for the genetic algorithm training. A negative set of approximately 700 URLs was also used to train the SVM. Strategies were evolved a different crawl lengths of 10000 URLs. The best performing strategies were tested with different seeds sets and crawls of length 10000, 20000, and 30000.

The genetic algorithm was run for 50 generations with a population size of 50, a crossover probability of 0.90, and mutation probability of 0.01. Each individual in the population is a bit string. The length of an individual is determined by allotting seven bits for each generation of parent pages evaluated plus seven bits for each of the hub, authority, community in, and community out scores. Each weight is calculated by calculating the decimal value of the seven bits allocated for that score to get a number between 0 and 127. This value is normalized so that the total value of all of the weights adds up to 1. The GA uses a modified version of the CHC elitist selection (Eshelman, 1991). In each generation the population size is doubled by creating new individuals. All of the individuals, both new and old, are sorted by fitness and the individuals with the highest fitness become the new population in the next generation. Individuals are selected for reproduction using the standard roulette wheel selection where individuals are chosen probabilistically based on their fitness. More fit individuals are more likely to be chosen for reproduction. A relatively high probability of crossover of 0.90 is used because of the elitist selection. The best individuals will be carried over into the next generation regardless of which individuals are chosen to reproduce, so current individuals do not need to be maintained by having a lower probability for crossover. One-point crossover was used as shown in figure 2. A crossover point, n, is chosen between 0 and the *length* of the individual with uniform probability. Each individual is cut at this point with the first n bits of each

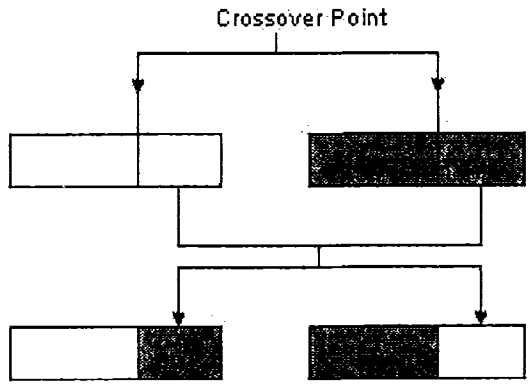

Figure 2. One point crossover

individual matched with the last $length - n$ bits of the other individual. If an individual is chosen for mutation, a bit is chosen with uniform probability and its current value of 0 or 1 is flipped.

3.1. Simulation: 10,000 URLs

The system was tested by evolving strategies to crawl 10000 URLs starting from the seed set of 139 URLs on the middle eastern dance data set. The number of levels used to calculate parent scores was set to five. Each strategy was tested by simulating three separate crawls and the resulting fitnesses were averaged together to get the overall fitness of the individual. The resulting five best individuals were then inspected and tested against the Best First strategy that corresponds to the P_1 weight of 1.00 and all other weights 0.00.

Table 1. Three best individual weights evolved on run of 10000 URLs.

Weight for Score Type	Best Over 10000
HUB	0.025828
$AUTHORITY$	0.025852
$COMM_{in}$	0.196382
$COMM_{out}$	0.312661
$PARENT_1$	0.250646
$PARENT_2$	0.093023
$PARENT_3$	0.080103
$PARENT_4$	0.005168
$PARENT_5$	0.010336

The weights evolved for the rank function for the best strategy are shown in Table 1. The evolved strategies give most weight to three scores, $COMM_{in}$, $COMM_{out}$, and the $PARENT_1$ scores, with the highest weight given to the $COMM_{out}$ score. The parent scores from higher generations are used in the rank function, but are given much lower weight than the direct parent page. The hubs and authorities scores are weighted low in the rank score, thus were not important in the strategy.

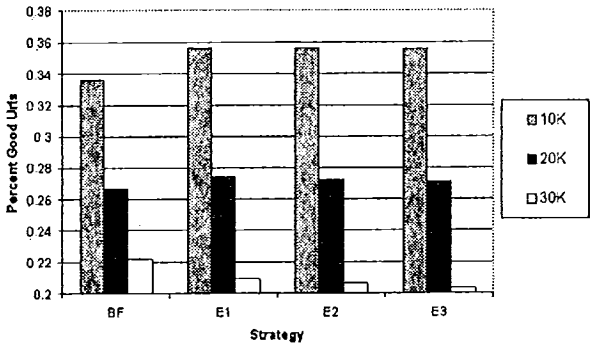

Figure 3. Fitnesses for middle eastern dance data

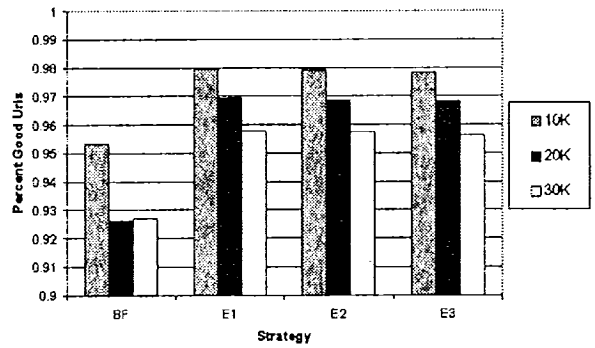

Figure 4. Fitnesses for baseball data

Each strategy was tested 15 times using a sequence of 15 random seeds . The fitness of each strategy was averaged over the 15 simulations. The simulation was run for 10000, 20000, and 30000 URLs on the two data sets using two different seed sets, a larger set of approximately 250 URLs, and a smaller set of about 50 URLs. Figure 3 shows the fitnesses of the Best First and the three best evolved strategies for the middle eastern dance data. The evolved strategies outperformed the Best First strategy for the 10000 and 20000 URL runs, but Best First performed better on the longer 30000 URL crawl. Figure 4 shows the fitnesses for the strategies when the simulations were run using the baseball data set. On this data, the three evolved strategies outperformed Best First on all of the runs. All of

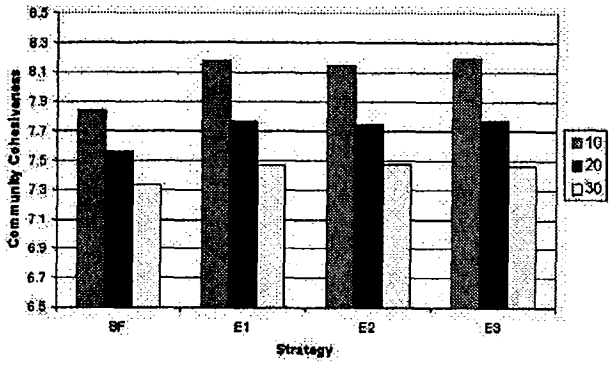

Figure 5, Community cohesiveness for middle eastern dance data

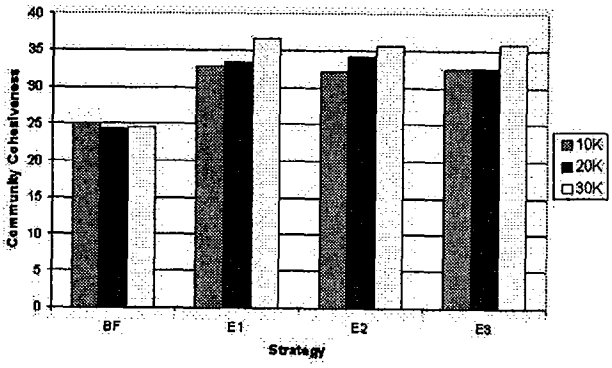

Figure 7. Community cohesiveness for baseball data

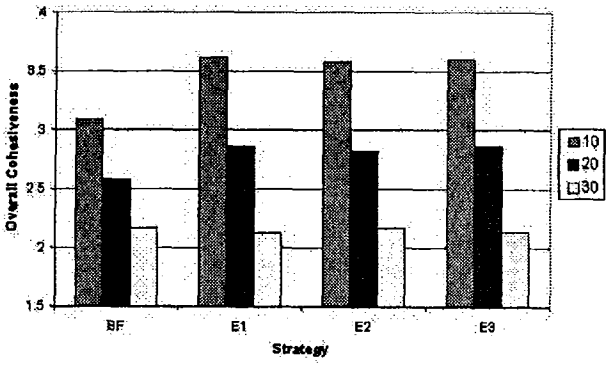

Figure 6. Overall cohesiveness for middle eastern dance data

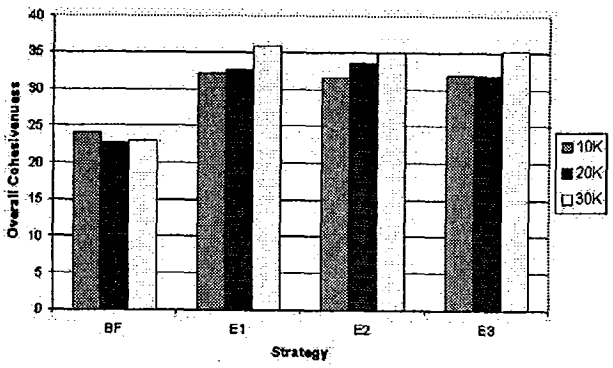

Figure 8. Overall cohesiveness for baseball data

the strategies performed better on this data than on the middle eastern dance data, because the data set is made up of a higher percentage of relevant pages.

We then looked at the link cohesiveness of two graphs built during the simulation, the community and the overall crawl. Cohesiveness is defined to be

$$Cohesiveness = \frac{\sum_{i=1}^{n} |L_i \cap C|}{|G|}, i \in G \qquad (13)$$

For community cohesiveness, G is the community, C, and for overall cohesiveness G is the set of all pages downloaded during the simulation. Figure 5 shows the community cohesiveness for the middle eastern dance data set. Figure 6 shows the overall cohesiveness for this data set. For both types, cohesiveness is higher for evolved strategies on the crawls of 10000 and 20000 and is very close to the Best First strategy on the crawl of 30000 URLs. This follows the behavior of the fitness on this data set with the difference between the evolved strategies and Best First growing smaller as the crawl size increases. Figure 7 shows the community cohesiveness for the baseball data set and Figure 8 shows the overall cohesiveness. The overall cohesiveness for this set is similar to the community because the set contains so many relevant pages. In this set, cohesiveness is much higher in the evolved strategies and increases as the size of the crawl increases. We expect that the evolved strategies will have higher cohesiveness because of the high weight given to the two community scores. The middle eastern dance community loses cohesiveness as the length of the crawl grows because it is a smaller community. As the crawl spreads out from the seed set, the number of relevant pages decreases rapidly.

Figure 9 shows a histogram of the number of links from the seed set the downloaded URLs are found on the baseball data set. The histogram shows that the evolved strategies download URLs closer to the original seed set than the Best First crawler. This again follows from the high weight given to the community scores. The reliance on the community scores would

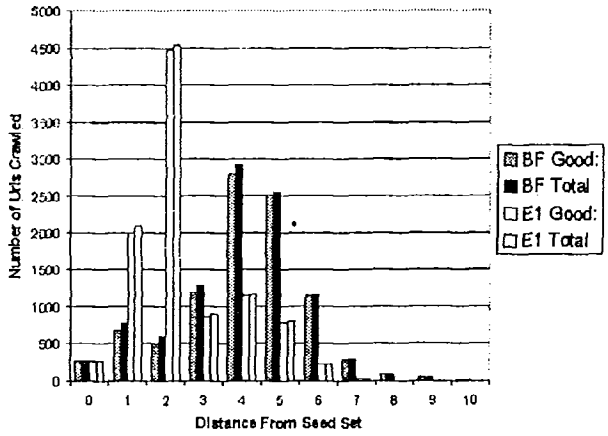

Figure 9. URLs crawled at distances for baseball data

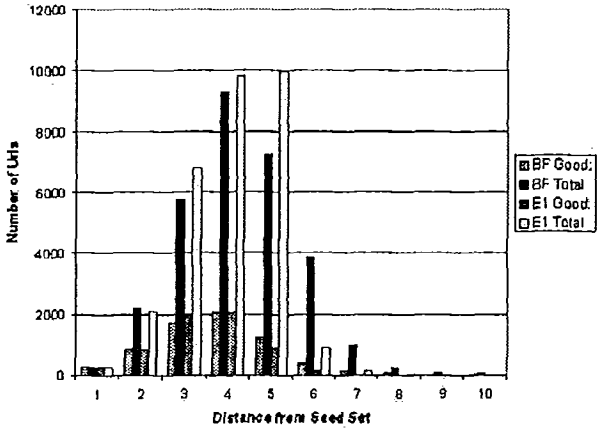

Figure 10. URLs crawled at distances for middle eastern dance data

account for this behavior by the evolved strategies. The $COMM_{out}$ score gives weight to URLs whose parents point into the community. As URLs are downloaded and added to the community, the ranks of the siblings of these URLs are increased. URLs that are at the same level become more likely to be chosen for downloading. Also the fitness function encourages this behavior by rewarding exploitation of the search space by only considering the number of good URLs found relative to the total URLs downloaded. The majority of URLs crawled and good URLs found by the evolved strategies are at level 1 and 2, or one to two links away from the seed set. In contrast, Best First finds most of its URLs in levels 4 and 5. Best First crawled out to level 14, while the maximum distance from the seed set crawled by the evolved strategies was 10. Thus the best first strategy crawls farther from the seed set and also finds URLs in a more distributed way than the evolved strategies.

The histogram for the middle eastern dance data shown in Figure 10 is similar. Again the evolved strategies download more pages closer to the seed set, this time mainly at levels 3, 4, and 5. The Best First strategies crawled to a maximum distance of 12 while the evolved strategies only crawled out to a distance of 10 links from the seeds. In this case the number of good URLs crawled was much smaller than in the case of baseball and were mostly found at 3 to 4 links from the seed set. Here the histograms for Best First and the evolved strategies were more similar to each other. This can be explained by the nature of the middle eastern dance data. Because the number of relevant pages is small in comparison with the size of the database, both strategies had a harder time finding good pages than with the baseball data.

4. Conclusions

In conclusion, the system was able to use the genetic algorithm to evolve strategies that combined analysis of text and link structure to outperform the Best First strategy on crawls of several different durations. One reason for this is that the evolved strategies are able to use link structure to differentiate between sibling links on the same page. So the evolved strategies are able to rank links on the same page differently based on how those URLs are linked to other pages in the crawl. The strategies evolved concentrated on exploiting the known good information more than on exploring new, possibly promising areas for search because of the nature of the fitness function used by the genetic algorithm. The fitness was only based on the percentage of good pages found. The evolved strategies are biased toward a greedy approach that tends to crawl URLs within a few links of the seed set. This worked well for a topic like baseball where many relevant pages are available. A fitness function that rewards exploration may help to find a strategy that is able to more effectively crawl smaller communities.

Future work will include evolving strategies using different fitness functions. A fitness function that rewards exploration may be able to evolve strategies that perform better for crawls of longer duration. Based on the intuition that differing strategies may perform better depending on the distance crawled from the seed set, evolving several sub strategies may be useful. Strategies that use different sub strategies at different points in the crawl may also find better overall strategies for longer crawls. Good short term substrategies might be found by initializing the population with

strategies found during previous runs on shorter length crawls. Injecting solutions to previously solved problems into a new population has been found to produce better solutions to new, similar problems in (Louis & Johnson, 1997). Other scores can be added to the rank score. The text in the neighborhood of a hyper link has been found to be a useful indicator of the topic of the linked page. A similarity measure of the extended anchor text to the topic can be added to the rank scores. Another indicator might be whether good URLs have been observed on the host site for the URL being considered.

References

Chakrabarti, S., Punera, K., & Subramanyam, M. (2002). Accelerated focused crawling through online relevance feedback. *WWW, Hawaii.* ACM.

Chakrabarti, S., van den Berg, M., & Dom, B. (1999). Focused crawling: a new approach to topic-specific Web resource discovery. *Computer Networks (Amsterdam, Netherlands: 1999), 31*, 1623–1640.

Diligenti, M., Coetzee, F., Lawrence, S., Giles, C. L., & Gori, M. (2000). Focused crawling using context graphs. *26th International Conference on Very Large Databases, VLDB 2000* (pp. 527–534). Cairo, Egypt.

Drucker, H., Shahrary, B., & Gibbon, D. C. (2001). Relevance feedback using support vector machines. *Proc. 18th International Conf. on Machine Learning* (pp. 122–129). Morgan Kaufmann, San Francisco, CA.

Eshelman, L. J. (1991). The chc adaptive search algorithm: how to have safe search when. engaging in nontraditional genetic recombination. In G. J. E. Rawlins (Ed.), *Foundations of genetic algorithms*, 265–283. Los Altos, CA: Morgan Kaufmann Publishers.

Flake, G., Lawrence, S., & Giles, C. L. (2000). Efficient identification of web communities. *Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 150–160). Boston, MA.

Gibson, D., Kleinberg, J. M., & Raghavan, P. (1998). Inferring web communities from link topology. *UK Conference on Hypertext* (pp. 225–234).

Goldberg, D. (1989). *Genetic algorithms in search, optimization and machine learning.* Reading, MA: Addison-Wesley.

google (2003). http://www.google.com/.

Holland, J. (1975). *Adaption in natural and artificial systems.* Ann Arbor, MI: University of Michigan Press.

Joachims, T. (1998). Text categorization with support vector machines: learning with many relevant features. *Proceedings of ECML-98, 10th European Conference on Machine Learning* (pp. 137–142). Chemnitz, DE: Springer Verlag, Heidelberg, DE.

Kleinberg, J. M. (1999). Authoritative sources in a hyperlinked environment. *Journal of the ACM, 46*, 604–632.

Louis, S. J., & Johnson, J. (1997). Solving similar problems using genetic algorithms and case-based memory. *Proceedings of the Seventh International Conference on Genetic Algorithms (ICGA97).* San Francisco, CA: Morgan Kaufmann.

Menczer, F., Pant, G., & Srinivasan, P. (2003). Topical web crawlers: Evaluating adaptive algorithms. http://dollar.biz.uiowa.edu/~fil/Papers/TOIT.pdf.

Menczer, F., Pant, G., Srinivasan, P., & Ruiz, M. E. (2001). Evaluating topic-driven web crawlers. *Research and Development in Information Retrieval* (pp. 241–249).

Pant, G., & Menczer, F. (2002). Myspiders: Evolve your own intelligent web crawlers. *Autonomous Agents and Multi-Agent Systems, 5*, 241–249.

Srinivasan, P., Menczer, F., & Pant, G. (2003). A general evaluation framework for topical crawlers. *Information Retrieval, Submitted.*