# Mining Top – $k$ Ranked Webpages Using Simulated Annealing and Genetic Algorithms

P. Deepa Shenoy[1], K.G. Srinivasa[1], A.O. Thomas, K.R. Venugopal[1], and L.M. Patnaik[2]

[1] Department of Computer Science and Engineering
University Visvesvaraya College of Engineering, Bangalore – 560001
shenoypd@yahoo.com, kgsrinivas@msrit.edu, achint@ieee.org
[2] Microprocessor Applications Laboratory, Indian Institute of Science, Bangalore
lalit@micro.iisc.ernet.in

**Abstract.** Searching on the Internet has grown in importance over the last few years, as huge amount of information is invariably accumulated on the Web. The problem involves locating the desired information and corresponding URLs on the WWW. With billions of webpages in existence today, it is important to develop efficient means of locating the relevant webpages on a given topic. A single topic may have thousands of relevant pages of varying popularity. Top - $k$ document retrieval systems identifies the top - $k$ ranked webpages pertaining to a given topic. In this paper, we propose an efficient top-$k$ document retrieval method (*TkRSAGA*), that works on the existing search engines using the combination of Simulated Annealing and Genetic Algorithms. The Simulated Annealing is used as an optimized search technique in locating the top-$k$ relevant webpages, while Genetic Algorithms helps in faster convergence via parallelism. Simulations were conducted on real datasets and the results indicate that *TkRSAGA* outperforms the existing algorithms.

## 1 Introduction

Data mining and web mining are emerging areas of immense interest for the research community. These two fields deal with knowledge discovery on the Internet. Extensive work is being carried out to improve the efficiency of existing algorithms and to devise new and innovative methods of mining the Web. Such efforts have direct consequences on e-commerce and Internet business models.

The Internet can be considered as a huge database of documents, which is dynamic in nature and results in an ever-changing chaotic structure. Search engines are the only available interface between the user and the web. It allows the user to locate the relevant documents in the WWW. A huge number of webpages may exist on any given topic in the order of $10^4$ to $10^6$. It becomes tedious for the user to sift through all the web pages found by the search engine to locate the documents of interest to the user.

S. Manandhar et al. (Eds.): AACC 2004, LNCS 3285, pp. 137–144, 2004.

The problem of page ranking is common to many web-related activities. The basic goal of ranking is, providing relevant documents on a given search topic. Top - $k$ selection queries are being increasingly used for ranking. In top - $k$ querying, the user specifies target values for certain attributes and does not expect exact matches to these values in return. Instead a ranked list of top - $k$ objects that best match the attribute values are returned [5].

Simulated Annealing (SA) is a powerful stochastic search method applicable to problems for which little prior knowledge is available. It can produce high quality solutions for hard optimization problems. The basic concept of SA comes from condensed matter physics. In this technique, the system (solid) is first heated to a high temperature and then cooled slowly. The system will settle in a minimum energy state if the cooling point of the system is sufficiently slow. This process can be simulated on a computer. At each step of the simulation, a new state of the system is generated from the current state giving a random displacement to a randomly selected particle. The new generated state will be accepted as the current state, if the energy of the new state is not greater than that of the current state. If not, it will be accepted with the probability, $e^{(-(E_{new-state}-E_{current-state})/T)}$, where E is the energy of the system and T is the temperature. This step can be repeated with a slow decrease of temperature to find a minimum energy state [1][3][4].

Another tested soft computing approach is Genetic Algorithms (GA), which works on the concept of evolution. Every species evolves in a direction suited for its environment. The knowledge they gain in this evolution is embedded in their chromosomal structure. The changes in chromosomes will cause changes in the next generation. The changes occur due to mutation and crossover. Crossover means the exchange of parts of genetic information between parents to produce the new generation. Mutation makes it possible for chromosomes to get a structure which is more suitable for the environment.

A combination of SA and GA is appropriate to the problems that place a premium on efficiency of execution, i.e., faster runtimes. This is an important consideration in any web-based problem as speed is of the utmost importance. The SA and GA techniques can be combined in various forms. GA can be applied before or after or even during the annealing process of the system under consideration [2].

Any page ranking algorithm has to be applied online and should be fast and accurate. The existing page ranking algorithms, though they give complete results, returns an enormous number of webpages resulting in lower efficiency. The use of soft computing approaches can give near optimal solutions, which are better than existing algorithms. In this paper, we combine Simulated Annealing with Genetic Algorithms to devise an efficient search technique. The Simulated Annealing is used because of its ability to handle complex functions and Genetic Algorithms is used to choose between the set of points in the intermediate states of Simulated Annealing, so as to eliminate the points that do not satisfy the fitness function. We thus achieve more accurate results with fewer runs of SA.

## 2 Problem Definition

Given a query to a search engine, returns a large number of web documents in terms of URLs (Uniform Resource Locators). Each webpage is characterized by the number of hits(the number of times a URL has been accessed by past users), number of referrer pages(incoming links), number of referred pages(out going links) and the number of occurances of the specified keywords of the given query. Let $E$ be the dataset containing the set of URLs and their corresponding characterstics, i.e. $E = \{U_m, S_m\}$, where $1 \leq m \leq n$ and $n$ is the total number of URLs returned. The function $S_m = N_m + I_m + O_m + D_m$, where, $N_m$ is the number of hits, $I_m$ is the number of incoming links, $O_m$ is the out going links and $D_m$ is the number of occurances of query keywords for the corresponding $m^{th}$ URL. Our objective is to find the top - $k$ relevant web documents from the dataset $E$ using combination of Simulated Annealing and Genetic Algorithms.

## 3 System Architecture

This section deals with the various modules involved in the system. The first step is to submit a query to a commonly used search engine. The query is a string or collection of strings that represent a set of keywords for a particular topic in which the search is being performed. Each string in the query is separated by a space or a special symbol. The query is represented as a set, $S = \{s_1, s_2, s_3, ...s_n\}$, $s_k$ is the $k^{th}$ string in the query. The query is submitted to the search engine. Once the search engine completes the search process, it will return a set of $n$ unique web documents (URLs). It can be represented as the set, $E = \{U_m, S_m\}$ where $1 \leq m \leq n$. $U_m$ is the actual address of $m^{th}$ URL in the result and $S_m$ is the function on URL $U_m$. The resulting URLs are categorized by their characterstic function $S_m$ to ease the retrieval process. Once the search engine returns $n$ URLs, an objective function over $S$ will be generated using harmonic analysis. The algorithm *TkRSAGA* is executed on the objective function *f(x)* and outputs the top - $k$ ranked URLs.

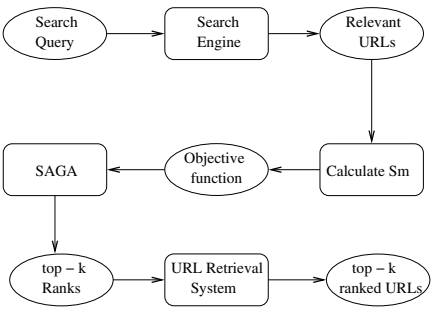


**Fig. 1.** The System Architecture

## 4 Algorithm *TkRSAGA*

*Top – k Document Retrieval using Simulated Annealing and Genetic Algorithms:*
**Step 1: Preprocessing:** Submit a query to an existing search engine like Google. The search engine returns a list of n URLs (webpages) of relevance to the topic. Each entry *E*, in the list of returned URLs must be composed of two entries {U,S}. Thus E = {U, S}, where $U$ is the actual URL and $S$ is the function over the corresponding to URL $U$ and is denoted as $\{(U_1, S_1), (U_2, S_2), ...(U_n, S_n)\}$.

**Step 2: Harmonic Analysis:** Let the output of Step 1 be denoted as $\{(n_1, s_1), (n_2, s_2), ...(n_n, s_n)\}$, where $n_m$ is the $m^{th}$ URL and $s_m$ is the function over $m^{th}$ URL and the objective function over these $n$ points can be generated using the formula $f(x) = a_0 + a_k cos(k\pi) + b_k sin(k\pi)$, where $1 \leq k \leq n$.

**Step 3: Performing Search:** The combination of Simulated Annealing and Genetic Algorithms can be applied over the objective function *f(x)* as given below,

**Algorithm:** Generate initial states $\alpha_0, \alpha_1, ...\alpha_{n-1}$ at random.
Generate initial temperature $T_0$.
loop
for each $\alpha_i$ in $\{\alpha_0, \alpha_1, ...\alpha_{n-1}\}$
loop
$\beta_i$ = generate_state($\alpha_i, T_j$);
until point $\alpha_i$ satisfies $\{curve \pm \epsilon\}$, where $\epsilon$ is the error,
if accept_state($\alpha_i, \beta_i, T_j$), then $\alpha_i = \beta_i$,
next $\alpha_i$,
for each $i$, $\{0 \leq i \leq n-2\}$
crossover_pairs($\alpha_i, \alpha_{i+1}$)
$\alpha_i$ = calculate_fitness($\alpha_i, \alpha_{i+1}$)
next $i$,
$T_{j+1}$ = update_state($T_j$),
$j = j + 1$;
until $k$ states remain.
**End**

Let the initial states $\alpha_0, \alpha_1, ...\alpha_{n-1}$ be a randomly chosen set of points from the objective function $f(\alpha)$, where $0 \leq \alpha_i \leq 2\pi$. The points $\alpha_i$ are chosen on the $x$ - axis at evenly spaced intervals. However, the actual initial states are computed usng the objective function *f(x)*. The Simulated Annealing technique cools the system uniformly and slowly from a higher initial temperature $T_0$ to a lower final temperature $T_k (T_0 > T_k)$. In the next iteration, a random state is generated by the function generate_state($\alpha_i, T_j$) and is determined by the probability $G_{\alpha\beta}(T_j)$ of generating a new state $\beta_i$ from an existing state $\alpha_i$ at temperature $T_j$. The generation function is defined as $g_i(Z) = 2 * (\mid Z \mid +1/ln(1/T_j)) * ln(1 + ln(1/T_j))$. The generation probability is given by $G_j(Z) = \frac{1}{2} + (\Sigma z * ln(1+ \mid z \mid ln(1/T_j)))/2 * ln(1 + ln(1/T_j))$.

The newly generated state $\beta_i$ is checked for acceptance by the function accept_state($\alpha_i, \beta_i, T_j$) and is determined by the probability $A_{\alpha\beta}(T_j)$ of accepting

state $\beta_i$ after it has been generated at temperature $T_j$. The acceptance probability $A_{\alpha\beta}(T_j)$ is given by, $A_{\alpha\beta}(T_j) = min\{1, exp(-(f(\beta) - f(\alpha))/T_j)\}$, where $f(\alpha)$ is the objective function considered for optimization. The new state $\beta_i$ is accepted only if it has lower energy state than the pervious state $\alpha_i$.

The rate of cooling in the Simulated Annealing technique(Annealing Schedule) is represented by $\rho$. It is a control parameter used to change the system temperature as the time progresses. The annealing schedule used in the algorithm is of the form, $T_k = T_0/e^{e^k}$, where $k$ represents the $k^{th}$ iteration. For practical considerations, the annealing schedule is set to $T_{n+1} = \rho T_n$. The function update_state($T_j$) updates the temperature with respect to the annealing schedule. The function crossover_pairs($\alpha_i, \alpha_{i+1}$) performs the genetic crossover operation on states $\alpha_i$ and $\alpha_{i+1}$. The random one-point crossover is performed on two states $i$ and $j$.

Finally, the function calculate_fitness($\alpha_i, \alpha_{i+1}$) performs the fitness calculation that is used to select the two states which are allowed to propagate to the next generation. The fitness function calculates the Euclidean distances of points $\alpha_i$ and $\alpha_{i+1}$ to the objective function *f(x)* and returns the closer point. Thus, the algorithm starts with an initial number of states and terminates with $k$ final states.

**Step 4:** Once the algorithm returns $k$ final states, they represent the points on the global minima over the objective function *f(x)*. These points can be mapped to the corresponding URLs and these URLs represent the top - $k$ ranked URLs.

## 5 Performance Analysis

The algorithm *TkRSAGA* works in two basic phases. The first phase involves the generation of the Fourier coefficients to determine the objective function *f(x)* and is linear with respect to the number of URLs supplied. The second phase is the application of combined SA and GA on the objective function *f(x)* to obtain the top - $k$ ranked URLs. The convergence of the second phase depends on the number of initial states, the annealing schedule and the initial temperature. Keeping these parameters constant for the test runs, we see that the performance curve for *TkRSAGA* tends to be linear. The execution tme is higher for smaller number of URLs and relatively lower for larger URLs. The graph of execution time versus the number of URLs for the algorithms *TkRSAGA* and HITS is shown in Figure 2(a). It shows that the algorithm *TkRSAGA* works better for larger databases.

The Figure 2(b), shows the graph of execution time versus the number of initial states and the performance curve is roughly logarithmic. As the number of initial number of states increases by a factor $x$, the execution time increases by a factor of *log(2x)*. This is obvious since, the initial states only influence the number of iterations made by GAs. After every crossover operation, exactly half the new generation is retained for future propagation. The graph in Figure 3(a), shows the execution time versus the desired top - $k$ ranks. The graph is plotted for varying number of URLs and varying $k$. Since the number of iterations increases for lower values of $k$, the curve is logarithmic.

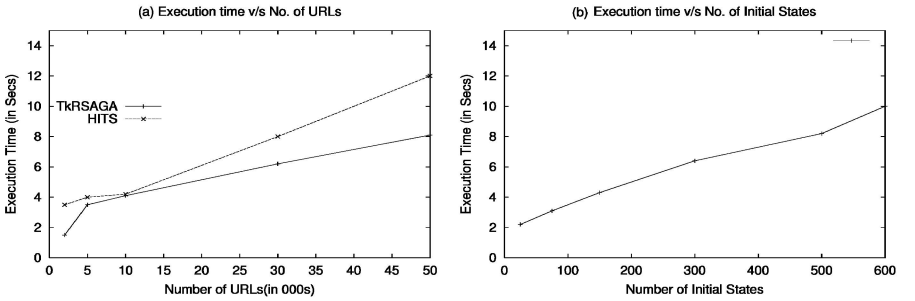


**Fig. 2.** 2(a): The graph of execution time versus number of URLs (Number of initial states = 128); 2(b): The graph of execution time versus number of initial states (Number of URLs = 10,000); for, Annealing schedule ($\rho$) = 0.95, k = 4

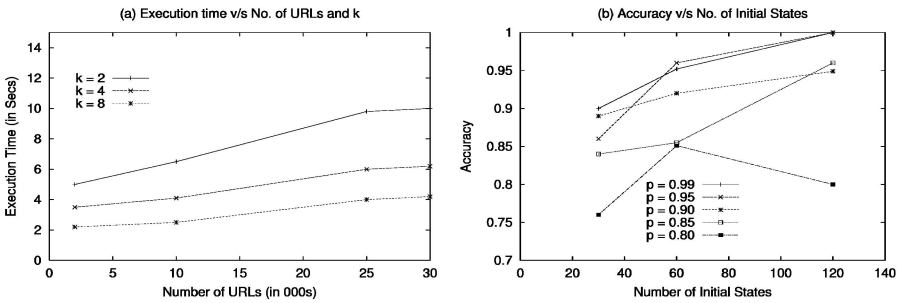


**Fig. 3.** 3(a): The graph of execution time versus varying number of URLs and $k$ (Number of initial states = 128 and Annealing schedule ($\rho$) = 0.95); 3(b): The graph of Accuracy versus number of initial states (Number of URLs = 10,000 and k = 4)

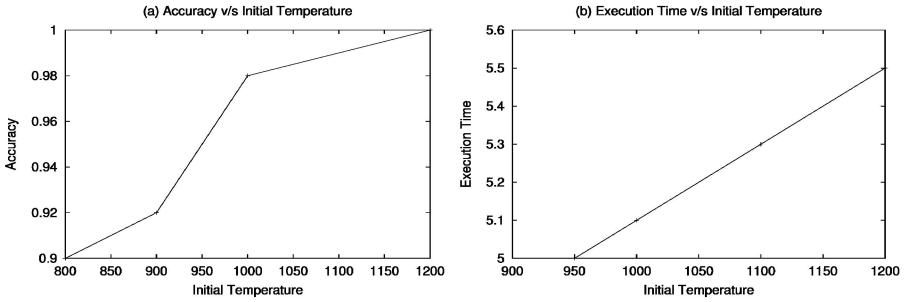


**Fig. 4.** 4(a): The graph of accuracy versus initial temperature; 4(b): The graph of execution time versus initial temperature; for, (Number of initial states = 128, Annealing schedule ($\rho$) = 0.95, Number of URLs = 10,000 and k = 4)

Figure 3(b), shows the graph of accuracy of the retrieved top - $k$ documents versus varying annealing scheduling ($\rho$) and the initial number of states. The accuracy parameter defines the ratio of the number of top - $k$ ranks returned

by the *TkRSAGA* to the desired top - $k$. The accuracy increases with the number of iterations. For higher values of initial states, better results are obtained. This is because the GAs produce the generations satisfying the fitness function. Similarly, for higher annealing schedules, the accuracy increases as SA performs more number of iterations in search of global optima.

The initial temperature $T_0$ determines the temperature of the system as it starts cooling. The higher the temperature, the more time it takes the system to reach the lower equilibrium state, i.e., the algorithm performs more number of iterations and takes longer time to reach the final k states. However, the number of iterations is directly proportional to the number of intermediate states being generated. Therefore, more the number of intermediate states, higher the accuracy and hence generates accurate $k$ final states. Thus, there exists a tradeoff between execution time and accuracy of results obtained, based on the initial temperature $T_0$. Figure 4(a), depicts the graph of initial temperature versus accuracy. Therefore, as the initial temperature increases, accuracy increases, in turn increasing the execution time. Figure 4(b), shows the linear relationship between the initial temperature and the execution time.

**Experiments on real datasets:** The datasets of university link files from cs.wlv.ac.uk are used for our experiments. A set of n webpages and corresponding number of hits are available. The number of hits is used to compute the harmonics for the objective function *f(x)*. The output of TkRSAGA is a set of $k$ values representing the top - $k$ relevant webpages. These values are mapped to the URLs to obtain the actual addresses. The HITS [5] algorithm is executed on the same database and the results of *TkRSAGA* and HITS algorithm are compared. The Table 1 shows the list of URLs and their corresponding number of hits. Table 2 shows the outputs of both *TkRSAGA* and HITS. The outputs

**Table 1.** Sample URLs taken from *cs.wlv.ac.uk*

| URL($U_m$) | No. of hits($N_m$) |
|---|---|
| www.canberra.edu.au/UCsite.html | 25482 |
| www.canberra.edu.au/secretariat/council/minutes.html | 1501 |
| www.canberra.edu.au/Staff.html | 199950 |
| www.canberra.edu.au/Student.html | 218511 |
| www.canberra.edu.au/crs/index.html | 178822 |
| www.canberra.edu.au/uc/privacy.html | 15446 |
| www.canberra.edu.au | 258862 |
| www.canberra.edu.au/uc/convocation/index.html | 16702 |
| www.canberra.edu.au/uc/staffnotes/search.html | 38475 |
| www.canberra.edu.au/uc/search/top.html | 190852 |
| www.canberra.edu.au/uc/help/index.html | 156008 |
| www.canberra.edu.au/uc/directories/index.html | 6547 |
| www.canberra.edu.au/uc/future/body.html | 25006 |
| www.canberra.edu.au/uc/timetable/timetables.html | 257899 |
| www.canberra.edu.au/uc/hb/handbook/search.html | 54962 |

**Table 2.** The output of *TkRSAGA* and HITS for ($T_0$ = 1200, No. of Initial States = 256, ($\rho$) = 0.95, $k$ = 4)

| RANK | *TkRSAGA* | HITS |
|---|---|---|
| 1 | www.canberra.edu.au | www.canberra.edu.au/ uc/timetable/timetables.html |
| 2 | www.canberra.edu.au/ uc/timetable/timetables.html | www.canberra.edu.au |
| 3 | www.canberra.edu.au/Student.html | www.canberra.edu.au/Student.html |
| 4 | www.canberra.edu.au/Staff.html | www.canberra.edu.au/Staff.html |

of both the algorithms are same and our algorithm *TkRSAGA* executes much faster than HITS algorithm. From Table 2, we can conclude that *TkRSAGA* outperforms the HITS in execution time without compromising with the accuracy of the results obtained.

## 6 Conclusions

In this paper, we have proposed an efficient algorithm *TkRSAGA*, for mining top - $k$ ranked web documents using the combination of Simulated Annealing and Genetic Algorithms. The ability of SA to solve harder problems and the combination of GA to reduce the number of iterations of SA and the inherent parallelism has made the algorithm efficient and effective.

## References

1. Xin Yao, "Simulated Annealing with Extended Neighbourhood", International Journal of Computer Mathematics, 40:169 - 189, 1991.
2. Xin Yao, "Optimization by Genetic Annealing", Proc. Second Australian Conference on Neural Networks, pp. 94 - 97, 1991.
3. H.H. Szu and R.L. Hartley, "Fast Simulated Annealing", Physics Letters, 122:157 - 162, 1982.
4. L. Ingber, "Very Fast Simulated Re-Annealing", Mathl. Comput. Modelling, 12(8):967 - 973, 1989.
5. J.M. Kleinberg, "Authoritative Sources in a Hyperlinked Environment", Proc. ACM - SIAM Symp. on Discrete Algorithms, 1998.