

Mining Top - k Ranked Webpages using Simulated Annealing and Genetic Algorithms

Srinivasa K. G*, Achinth O Thomas
P. Deepa Shenoy**, Venugopal K. R**, L. M. Patnaik***

5th May 2004

Abstract

Searching on the Internet has grown in importance over the last few years, as huge information is invariably accumulated on the Web. The problem involves in locating the desired information and corresponding URLs on 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 and 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 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.

Keywords: Web Mining, Page Ranking, Simulated Annealing, Genetic Algorithms, SAGA

* Contact Author: Email: kgsrinivas@msrit.edu

** Department of Computer Science and Engineering, University Visvesvaraya College of Engineering, K.R.Circle, Bangalore - 560001. Email: vkrajuk@vsnl.com

*** Microprocesor Applications Laboratory, Indian Institute of Science, Bangalore. Email: lalit@micro.iisc.ernet.

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 ever-changing chaotic structure. Web mining can be of three types, Web Content mining (WCM), Web Structure mining (WSM) and Web Usage mining (WUM) [12]. WCM deals with the discovery of useful information from the Web contents / data / documents / services. Web contents need not be only text; they encompass a broad range of data such as audio, video, symbolic, metadata and hyper linked data. Of these, text and hypertext contents are of interest currently. WSM deals with mining the structure of hyperlinks within the Web itself. Here the structure represents the graph of the links in a site or between sites. WSM reveals more information than just the information contained in documents. Googles' Page Rank algorithm uses this approach. WUM mines the secondary data generated by the users' interaction on the Web. Web usage data includes data from web-server access

logs, proxy server logs, browser logs, user profiles, user sessions or transactions, user queries, bookmark folders, mouse clicks and scrolls on other types of generated data. WUM is primarily used by e-commerce companies for tracking customer behavior on their site and plays a major role in personalizing the web [12], [14].

Search engines are the only available interface between the user and web. It allows the user to locate the relevant documents in 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.

At present the page ranking algorithms like Google PageRank rank the web documents based on number of hits, number of links that point to that page, etc. and the search engine returns the pages in decreasing order of their rank. Page ranking is a commonly used approach in indexing the most relevant web pages on a topic. Primitive methods of ordering, like ranking, are giving way to more intuitive approaches such as taking into account the relative popularity of webpages and authorities of webpages that link to other pages. Webpages that have been ranked also need to be searched since, storage is not based on ranking. Linear searches are extremely inefficient for enormous number of webpages. The exponentially faster binary search is not feasible since pages need not be arranged in non-increasing/decreasing order. These considerations call for more advanced search techniques. The field of soft computing offers optimized searching techniques as the database expands rapidly.

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 [4].

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].

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 embedding 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 [3] [13].

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.

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, they produce 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 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.

This paper is organized as follows. Section 2 presents a review of related work. The problem of finding top - k ranked webpages using SAGA is addressed in section 3. Numerical results are analysed in section 4. Section 5 gives the conclusions.

1.1 Contributions of this paper

In this paper, we address the problem of locating top - k relevant documents on the WWW using the combination of Simulated Annealing and Genetic Algorithms. The use of Simulated Annealing ensures accurate results and finds a global minima for any complex functions. The Genetic Algorithms is used for faster convergence and hence the reduced space and time complexity.

2. Related Work

Considerable interest has been generated in the field of web-related ranking and searching. Soft computing methods like Simulated Annealing and Genetic Algorithms are used tackle the enormity of the Internet. The analogy between finding minimum energy states in a physical system and finding minimum cost configurations in a combinatorial optimization problem is observed in [5]. A new SA algorithm for solving combinatorial optimization problems, based on the Metropolis procedure is proposed in [6]. A general model of SA with both dynamic generation and acceptance probabilities is addressed in [1]. A Fast Simulated Annealing Algorithm (FSA), Very Fast Simulated Annealing Algorithm (VFSA) and a New Simulated Annealing Algorithm (NSA) are given in [7], [8] and [2] respectively. The combination of Simulated Annealing and Genetic Algorithms to improve the solution to a hard combinatorial optimization problem called Genetic Annealing is discussed in [3].

In the area of webpage ranking, the Hyperlink Induced Topics Search (HITS) algorithm [9] and Google's PageRank [10] are widely applied to analyze the structure of the Web. HITS estimates the authority and hub values of hyperlinked pages in the Web. PageRank merely ranks pages. The HITS algorithm has to construct the search space as a graph. This leads to a higher space and time complexity as database increases and the algorithm cannot be implemented on a distributed architecture. The FDS algorithm [11] takes advantage of spatial information to give a more accurate ordering of relevance to a document set. The algorithms in [10] and [11] retrieve all the relevant documents for a given topic, inturn increasing the time complexity.

Extensive work has been done in the field of using Simulated Annealing for function optimization. However, the use of SA alone to optimize a function has its drawbacks. The speed of convergence of the algorithm is very slow. This is due to the very nature of the annealing process, which has to always maintain the system in a state of equilibrium.

3. Problem Definition

Given a query to a search engine, results in a large number of relevant 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 characteristics, 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 is defined as, $S_m = N_m + I_m + O_m + D_m$, 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 occurrences 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 the combination of Simulated Annealing and Genetic Algorithms.

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 different string in the query is separated by a space or a special symbol. The query is represented as a set, $S = \{s_1, s_2, s_3, \dots, s_n\}$, 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 relevant unique web documents (URLs). It can be represented as the set, $E = \{U_m, S_m\}$, where $1 \leq m \leq n$. U_m actual address of m^{th} URL in the result and S_m is the function on URL U_m .

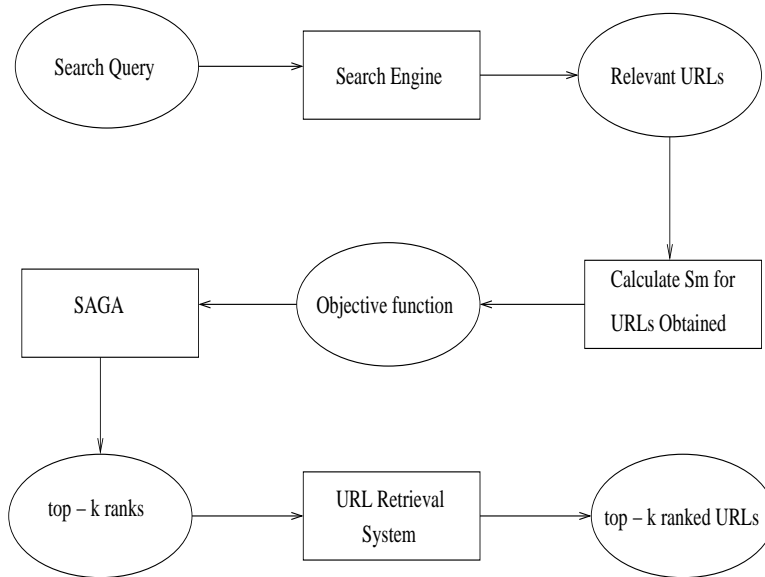


Figure 1: The System Architecture

The resulting URLs are categorized by their characteristic function S_m to ease the retrieval process. Once the search engine returns n relevant 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.

Algorithm

Top - k document Retrieval using Simulated Annealing and Genetic Algorithms (*TkRSA GA*):

Step 1: Preprocessing: Submit a query to an existing search engine like Google. The search engine returns a list of n URLs (web pages) of relevance to the topic. Each entry E , in the list of returned URLs must be composed of two entities $\{ U, S \}$. 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 $\{ (U_1, S_1), (U_1, S_2), \dots (U_n, S_n) \}$.

Step 2: The harmonic analysis: Consider a function $f(x)$ specified through a table of values of x and $y = f(x)$. According to the property of definite integrals, the mean value of $\phi(x)$ over the interval (a, b) is given by

$$[[\phi(x)]] = \frac{1}{b-a} \int_a^b \phi(x) dx$$

In Fourier mathematics, the Euler formulae for Fourier co-efficients are given as, with the integral limits ranging from a to $a+2l$.

$$a_0 = (1/l) \int f(x) dx = 2[[f(x)]]$$

$$a_n = (1/l) \int f(x) \cdot \cos\left(\frac{h\pi}{l}\right)x dx, \\ = 2[[f(x) \cdot \cos\left(\frac{h\pi}{l}\right)x]].$$

$$b_n = (1/l) \int f(x) \cdot \sin\left(\frac{h\pi}{l}\right)x dx, \\ = 2[[f(x) \cdot \sin\left(\frac{h\pi}{l}\right)x]], \text{ Where } n = 1, 2, 3, \dots$$

Here $[[\phi(x)]]$ is the mean value of $\phi(x)$ over interval $(a, a+2l)$, which is of length $2l$.

The Fourier expansion of a function (objective function) $f(x)$ is given as,

$$f(x) = \frac{a_0}{2} + \sum a_n \cdot \cos\left(\frac{h\pi}{l}\right)x + \sum_{n=1}^{\infty} b_n \cdot \sin\left(\frac{h\pi}{l}\right)x. \text{ Where } (1 \leq n \leq \infty)$$

In this expansion of $f(x)$, the term $(a_1 \cos(x) + b_1 \sin(x))$ is called the fundamental harmonic. Since $f(x)$ is the summation of series of harmonics, it is thus possible to break a function $f(x)$ into a series of sine and cosine terms. Using this technique, we can convert a table of values of x and $y = f(x)$ into a function composed of *sine* and *cosine* terms.

Let the output of **Step 1** be denoted as $\{ (n_1, s_1), (n_2, s_2), \dots (n_n, s_n) \}$, 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 following algorithm.

1. Initialization

$f_len = n$; {number of results returned by search engine}

$\theta = 2\pi / f_len, count = 1$;

$k = 0$;

2. Generation of Objective Function

While (not eof)

read n_m from the file;

$a_0 = a_0 + s_m$;

for each count from 0 to 7

$a[count - 1] += s_m * \cos[count * k]$;

$a[count - 1] += s_m * \sin[count * k]$;

end for

$k += \theta$;

end while

3. For each i ($1 \leq i \leq n$), $a[i] = b[i] / = 2\pi / \theta$.

The function $f(x)$ is defined as, $f(x) = a_0 + a_k \cos(k\pi) + b_k \sin(k\pi), \{1 \leq k \leq n\}$.

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

```

Generate initial states  $\{\alpha_0, \alpha_1, \dots, \alpha_{n-1}\}$  at random.
Generate initial temperature  $T_0$ .
loop
  for each  $\alpha_i$  in  $\{\alpha_0, \alpha_1, \dots, \alpha_{n-1}\}$ 
    loop
       $\beta_i = \text{generate\_state}(\alpha_i, T_j)$ ;
      until point  $\alpha_i$  satisfies  $\{\text{curve} \pm \epsilon\}$ , where  $\epsilon$  is the error
      if  $\text{accept\_state}(\alpha_i, \beta_i, T_j)$ , then  $\alpha_i = \beta_i$ ,
    next  $\alpha_i$ ,
  for each  $i \{0 \leq i \leq n-2\}$ 
     $\text{crossover\_pairs}(\alpha_i, \alpha_{i+1})$ 
     $\alpha_i = \text{calculate\_fitness}(\alpha_i, \alpha_{i+1})$ 
  next  $i$ ,
   $T_{j+1} = \text{update\_state}(T_j)$ 
   $j = j+1$ ;
until  $k$  states remain.

```

Let the initial states $\{\alpha_0, \alpha_1, \dots, \alpha_{n-1}\}$ be randomly chosen set of points from the objective function $f(\alpha)$, where $0 \leq \alpha_i \leq 2\pi$. The points α_i are chosen on the x - axis at evenly spaced intervals. However, the actual initial states are computed using 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 $\text{generate_state}(\alpha_i, T_j)$ and is determined by the probability $G_{\alpha\beta}(T_j)$ of generating a new state β_i from an existing state α_i at temperature T_j . The generation function is defined as $g_i(Z) = 2^{*(|Z|+1/\ln(1/T_j))*\ln(1+\ln(1/T_j))}$. The generation probability is given by $G_j(Z) = 1/2 + (\Sigma z * \ln(1+|z|\ln(1/T_j))) / (2*\ln(1+\ln(1/T_j)))$. The newly generated state β_i is checked for acceptance by the function $\text{accept_state}(\alpha_i, \beta_i, T_j)$ and is determined by the probability $A_{\alpha\beta}(T_j)$ of accepting state β_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 β_i is considered for acceptance only if it has a lower energy state than the previous state α_i .

The rate of cooling of the Simulated Annealing Technique (Annealing Schedule) is represented by ρ . 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 $\text{update_state}(T_j)$ updates the temperature with respect to the annealing schedule. The function $\text{crossover_pairs}(\alpha_i, \alpha_{i+1})$ performs the genetic crossover operation on states α_i and α_{i+1} . The random one - point crossover is carried on two states i and j .

Finally the function $\text{calculate_fitness}(\alpha_i, \alpha_{i+1})$ 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 α_i and α_{i+1} to the objective function $f(x)$ and returns the closer point. Thus the algorithm starts with few 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.

4. 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 time 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. It shows that the algorithm *TkRSAGA* works better for larger databases. Since we are interested in very large databases, *TkRSAGA* works more efficiently than other algorithms.

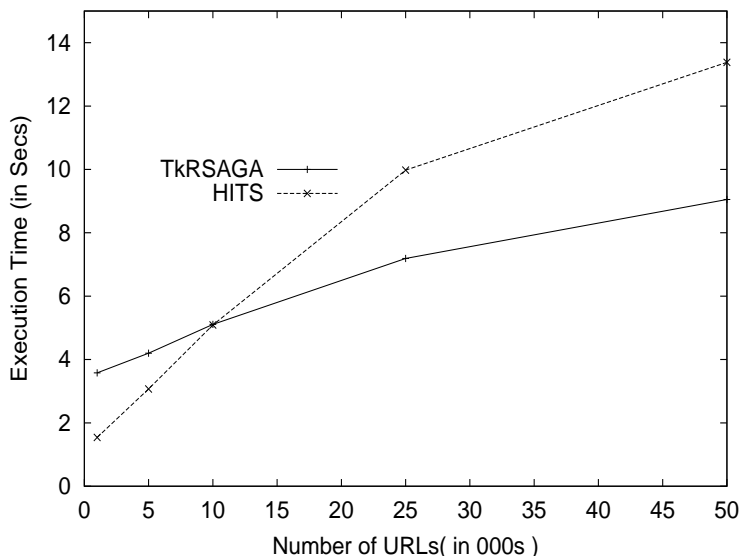


Figure 2: The graph of execution time versus number of URLs (Annealing schedule (ρ) = 0.95, Number of initial states = 128, $k = 4$)

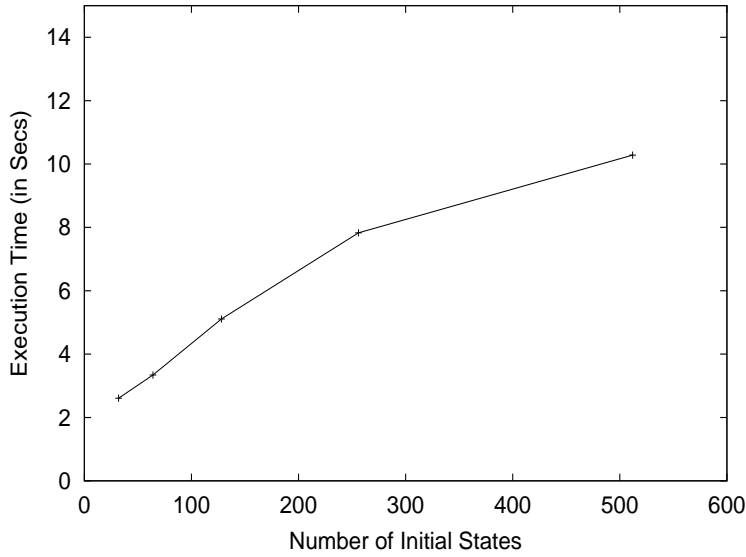


Figure 3: The graph of execution time versus number of initial states (Annealing schedule (ρ) = 0.95, Number of URLs = 10,000, k = 4)

The Figure 3, 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 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 4, 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.

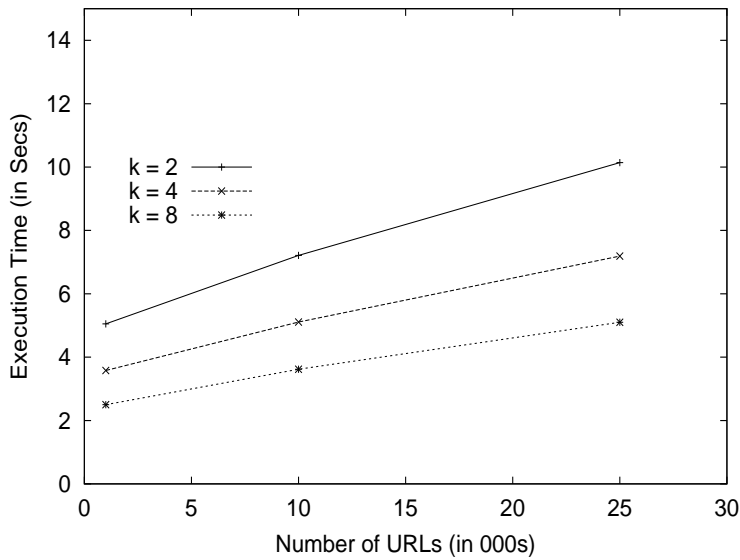


Figure 4: The graph of execution time versus varying number of URLs and k (Annealing schedule (ρ) = 0.95, Number of initial states = 128)

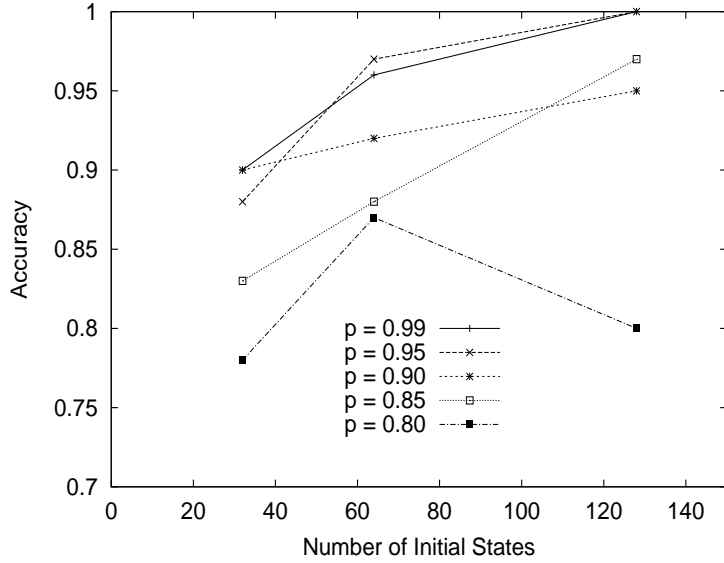


Figure 5: The graph of accuracy versus number of initial states
(Number of URLs = 10,000, $k = 4$)

Figure 5, shows the graph of accuracy of the retrieved top - k documents versus varying annealing scheduling(ρ) 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 that produces 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 propotional 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 6, depicts the graph of initial temperature versus accuracy. Therefore, as the initial temperature increases accuracy increases, inturn increasing the execution time. Figure 7, shows the linear relationship between the initial temperature and the execution time.

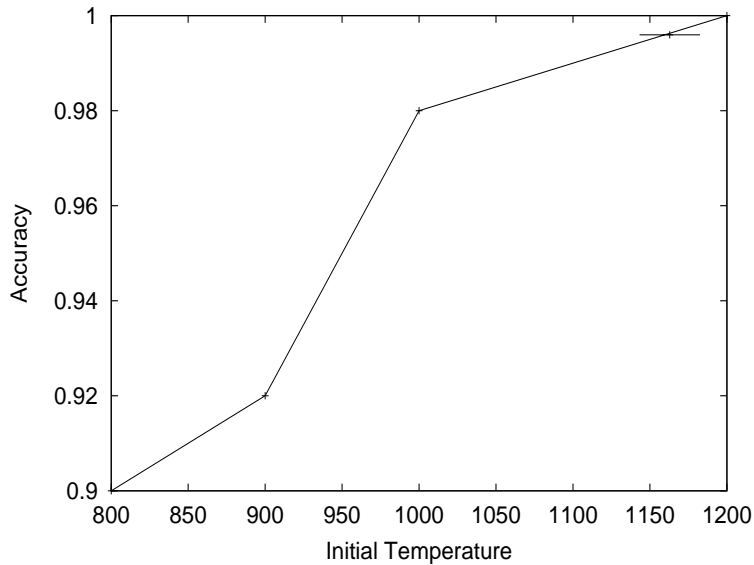


Figure 6: The graph of accuracy versus initial temperature
($k = 4$, Number of URLs = 10000, $\rho = 0.95$, Number of initial states = 128)

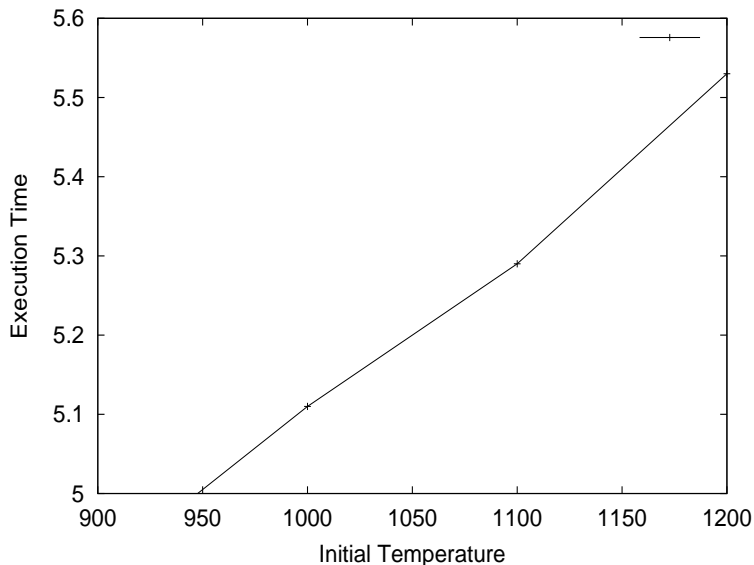


Figure 7: The graph of execution time versus initial temperature
($k = 4$, Number of URLs = 10000, $\rho = 0.95$, Number of initial states = 128)

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 objective function $f(x)$ is presented as input to the *TkRSAGA* algorithm along with the values for parameters like initial temperature, annealing schedule, initial number of states and number of top- k ranks required as output. When the *TkRSAGA* algorithm terminates, its output 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 [9] algorithm is executed on the same

database and the results of *TkRSAGA* and HITS algorithms are compared. The Table 1 shows the list of URLs and their corresponding number of hits. Table 2 and Table 3 shows the outputs of *TkRSAGA* and HITS respectively. The outputs of both the algorithms are same and our algorithm *TkRSAGA* executes much faster than HITS algorithm. From Table 2 and Table 3, we can conclude that *TkRSAGA* outperforms the HITS in execution time without compromising with the accuracy of the results obtained.

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 1: Sample URLs taken from cs.wlv.ac.uk/

URLs
www.canberra.edu.au
www.canberra.edu.au/uc/timetable/timetables.html
www.canberra.edu.au/Student.html
www.canberra.edu.au/Staff.html

Table 2: The output of *TkRSAGA*
($T_0 = 1200$, No. of initial states = 256, $\rho = 0.95$, $k = 4$)

URLs
www.canberra.edu.au/uc/timetable/timetables.html
www.canberra.edu.au
www.canberra.edu.au/Student.html
www.canberra.edu.au/Staff.html

Table 3: The output of HITS

5 Conclusions

Efficient retrieval of top - k web documents from WWW is an important task with respect to exponentially growing webpages. 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 itera-

tions of SA and the inherent parallelism has made the algorithm efficient and effective. Our algorithm is better than the existing algorithms both with respect to time complexity and space complexity. The algorithm can be tested on a distributed environment.

References

- [1] Xin Yao, Simulated Annealing with Extended Neighbourhood in International Journal of Computer Mathematics, 40:169 - 189, 1991.
- [2] Xin Yao, A New Simulated Annealing Algorithm in International Journal of Computer Mathematics, 56:161 - 168, 1995.
- [3] Xin Yao, Optimization by Genetic Annealing in Proc. Second Australian Conference on Neural Networks, ed. M. Jabri, pp. 94 - 97, 1991, Sydney, Australia.
- [4] Kaushik Chakrabarti, Michael Ortega-Binderberger, Sharad Mehrotra and Kriengkrai Porkaew, Evaluating Refined Queries in Top-k Retrieval Systems in IEEE Transactions on Knowledge and Data Engineering, pp. 256 - 270, Vol. 16, No. 2, February 2004.
- [5] S. Kirkpatrick, C.D. Gelatt and M.P. Vecchi, Optimization by Simulated Annealing in Science, 220:671 - 680, 1983.
- [6] V. Cerny, Thermodynamical Approach to the Travelling Salesman Problem: An Efficient Simulation Algorithm in Journal of Optimization Theory and Applications, 45:41 - 51, 1985.
- [7] H.H. Szu and R.L. Hartley, Fast Simulated Annealing in Physics Letters A, 122:157 - 162, 1982.
- [8] L. Ingber, Very Fast Simulated Re-Annealing, in Mathl. Comput. Modelling, 12(8):967 - 973, 1989.
- [9] J.M. Kleinberg, Authoritative Sources in a Hyperlinked Environment, in Proc. ACM-SIAM Symp. Discrete Algorithms, 1998.
- [10] S. Brin and L. Page, The Anatomy of a Large-Scale Hypertextual Web Search Engine, in Proc. Seventh World Wide Web Conf., 1998.
- [11] Hung-Yu Kao, Shian-Hua Lin, Jan Ming Ho and Ming-Syan Chen, Mining Web Informative Structures and Contents Based on Entropy Analysis, in IEEE Transactions on Knowledge and Data Engineering, Vol. 16, No. 1, January 2004.
- [12] Sankar K. Pal, Varun Talwar and Pabitra Mitra, Web Mining in Soft Computing Framework: Relevance, State of the Art and Future Directions, in IEEE Transactions on Knowledge and Data Engineering.
- [13] P. Deepa Shenoy, Srinivasa K. G, Venugopal K. R, and L. M Patnaik, "Evolutionary Approach for Mining Association Rules on Dynamic Databases," *Proc. PAKDD 2003, Conf. , LNAI 2637*, 2003, pp. 325 - 336.
- [14] P. Deepa Shenoy, Srinivasa K. G, Venugopal K. R, and L. M Patnaik, "Hash Mine: An Efficient Web Log Mining Algorithm to Discover Interesting Access Patterns," *In Proc. ITPC 2003, Conf. , 2003*, pp. 160 - 167.