SEAL'02: PROC. OF THE 2002 ASIA-PACIFIC CONF. ON SIMULATED EVOLUTION AND LEARNING (NOV. 2002) PP. 208–213.

APPLYING EVOLUTIONARY COMPUTATION METHODOLOGIES FOR SEARCH ENGINE DEVELOPMENT

Reginald L. Walker

Computer Science Department
University of California at Los Angeles
Los Angeles, California 90095-1596
rwalker@cs.ucla.edu

ABSTRACT

Early search engines had their origin in information retrieval systems. These system typically were developed by human editors to index a document set that was static over long periods of time. The static document set provided a stable user environment that was optimized over time and facilitated incremental growth in the document collection. Early search engines used this tried and tested information retrieval model, but encountered usability limitations as the document growth rate accelerated. The limitations of the model became magnified as the need for automated indexing mechanisms grew, and information retrieval systems began to be used with dynamic document datasets. These limitations are still apparent in current search engines which incorporate aspects of these early information retrieval systems. This paper presents the Tocorime Apicu approach for replacing the information retrieval model with an information sharing model that adapts to changing conditions within the Internet using the stochastic optimization methodologies of evolutionary computation.

1. INTRODUCTION

The indexing mechanisms of search engines were developed as an extension of information retrieval (IR) systems [8, 15] designed to be employed to index bibliographic databases. A common restriction of most IR systems was that they indexed only document titles or abstracts, as opposed to the text of a document—although some IR systems did "full-text search." These early systems were known as document text retrieval systems, and encompassed the following two components of current engines: 1) indexing—the retrieval mechanism needed to match expressions of the user's information need (query) with the items in selected files (documents and/or Web pages), and 2) searching—the mechanism that matches search query items with files.

The classification aspects of IR systems are based on viewing document retrieval as a dynamic process involving learning in response to training examples and feedback. The basic components of an IR system—files (documents or Web pages), index terms (search strings), and user requests (queries)—reflect that: 1) the indexing mechanism has the ability to classify/reclassify files which are placed in pre-indexed stores of information, and 2) the searching mechanism functions as a parser with the ability to distinguish between matching and non-matching components within files.

2. REPLACING THE INFORMATION RETRIEVAL MODEL WITH AN INFORMATION SHARING MODEL

The social structure associated with honeybee behavior [6] provides a simple solution to the problem of multiway rendezvous [2]. The organizing principle of honeybee society [7] is information sharing (IS)—the ability to send and receive messages, and to encode and decode information that may be transmitted through chemical, tactile, auditory, and/or visual messages for the purpose of finding and communicating the location [10] of adequate food sources, including water.

Adaptation of the honeybee IS model establishes order in the inherent interactions between the search engine's indexer, Web foraging, and browser mechanisms by including the social (hierarchical) structure and simulated behavior of this complex system. The simulation of behavior will engender mechanisms that are controlled and co-ordinated in their various levels of complexity.

3. OPTIMIZATION OF HONEYBEE SEARCH STRATEGIES FOR A BETTER WEB IS SYSTEM

3.1. Optimization Techniques

The stochastic optimization techniques of evolutionary computation (EC) [3, 5] contain mechanisms which enable the representation of certain, unique aspects of honeybee be-

Table 1: EC methodologies that form the basic attributes of Tocorime Apicu engine.

EC method	**Usage**	**Mutation and/or recombination**	**Selection (Mating selection)**
Evolutionary strategies (ISI System)	Supersedure emulation extends the representation of individuals to include strategy parameters for adaptive selection and recombination rate(s).	Random speciation seed provides the selection features in the form of a probe set.	Tournament selection and proportional fitness selection are used.
Evolutionary programming (ISI System)	Supersedure emulation acts as a finite state machine (FSM) or automaton.	Recombination and mutation operators are used to change the states of the FSM.	Employs fitness-based recombination selection of multiple individuals and mutation [3]
Genetic algorithms (ISI System)	Web page clustering system uses chromosomes of fixed length as a page data structure.	Heavy use is made of recombination and mutation.	Fitness-proportional and random selection of individuals are used.
Classifier systems (ISI System) (HRD System)	Web page parsers operate as general purpose Turing complete algorithms comprised of production rules (HTML) as push-down automata.	Rules resulting from the application of a genetic algorithm build the next potentially modified classifier system.	The first pass parser validates the structure of each retrieved Web page by a HRD Web forager. The second strips unneeded information from each page provided by a Web forager.
Evolvable hardware (HRD System)	Web mechanisms used a reconfigurable hardware methodology (active networks) to facilitate a network probe (signaling) facility to query networks for HTML resource discovery, diagnostics, network monitoring, etc.	Computational measures are used to access network bandwidth and CPU requirements.	
Genetic programming (ISI System)	Web page clustering system uses chromosomes of fixed length as a page data structure with syntax wrappers.	Heavy use is made of recombination and mutation.	Fitness-proportional and random selection of individuals are used.

havior. The chief differences among the various types of EC stem from: 1) the representation of solutions (known as individuals in EC), 2) the design of the variation operators (mutation and/or recombination—also known as crossover), and 3) selection mechanisms. A common strength of these optimization approaches lies in the use of hybrid algorithms which are derived by combining one or more of the evolutionary search methodologies. These methodologies can be related to meaningful representation and effective matching of user needs with relevant documents. An individual in this research effort is considered a node/computer.

3.2. Use of the Optimization Methodologies

Table 1 presents the mapping of the optimization methodologies to the components of the Tocorime Apicu architecture. The HTML Resource Discovery (HRD) system [16] uses the methodologies of EHW and active networks (AN) in its network probing faculty to establish customized routes for the retrieval of remotely located Web pages by avoiding network congestion [11, 12, 13]. Certain aspects of EHW are also components of AN. The HRD system Web probes, scouts, and foragers are responsible for retrieving external data using honeybee foraging strategies.

Production rules are used to verify the format of raw pages retrieved by the HRD system as a first pass, and the information sharing indexing (ISI) system Web document parser [18] performs a second pass. GA and GP have been combined to provide the basis of the ISI indexing strategy. Individual representations in the ISI system are hash table data structures of fixed length with syntax wrappers. Recombination and mutation are applied frequently. Workload redistribution/load balancing is controlled via stochastic optimization techniques that incorporate GA, ES, EP, and GP. The selection operator uses fitness-proportional and random mechanisms to implement a nearest neighbors (NN) strategy [18]. The Web page dispatcher uses a stochastic regulatory mechanism to adaptively form clusters of indexer nodes—sets of nearest neighbors—that facilitate the migration of Web pages using the honeybee information sharing model. The ISI system uses probe sets which may be determined statically or dynamically, a set of retrieval algorithms, and selection methodologies of evolutionary computation.

4. EXPERIMENTAL RESULTS

4.1. HRD Results

4.1.1. HRD Experimental Environment

The goal of this study was to test the run-time environment associated with probe dispatchers and determine the limitations in executing the HRD network probing software for

Table 2: Cumulative access log summary for all Web probe dispatchers per week, starting on 15 Oct 2001 and terminating 07 Jan 2002.

Duration — Start date — Stop date	**Result code**	**Web probe dispatchers**					**Target ratio**
		Node 0	**Node 1**	**Node 2**	**Node 3**	**Totals**	
15 Oct - 22 Oct	W	856898	866100	937825	860185	3521008	88.03%
	X	1682	1197	1550	1114	5543	N/A
	Y	594	460	623	436	2113	N/A
22 Oct - 29 Oct	W	1554957	1590259	1650506	1579525	6375247	79.69%
	X	2987	2274	2539	2013	9813	—
	Y	986	878	1011	758	3633	—
29 Oct - 05 Nov	W	2316420	2353539	2411958	2360721	9442638	78.69%
	X	4535	3345	3649	2972	14501	—
	Y	1584	1328	1485	1140	5537	—
05 Nov - 12 Nov	W	3154581	3156620	3214533	3176032	12701766	79.39%
	X	6184	4391	4715	3940	19230	—
	Y	2247	1791	1899	1516	7453	—
12 Nov - 19 Nov	W	3938617	4008097	4025989	3976145	15948848	79.74%
	X	7624	5547	5937	4855	23963	—
	Y	2774	2252	2370	1857	9253	—
19 Nov - 26 Nov	W	4709028	4810252	4814126	4772524	19105930	79.61%
	X	9009	6625	7089	5919	28642	—
	Y	3164	2623	2773	2254	10814	—
26 Nov - 03 Dec	W	5490017	5576071	5595341	5560066	22221495	79.36%
	X	10525	7798	8190	6919	33432	—
	Y	3614	3042	3148	2568	12372	—
03 Dec - 10 Dec	W	6308515	6405952	6377555	6369867	25461889	79.57%
	X	12101	8935	9365	7970	38371	—
	Y	4228	3480	3606	2940	14254	—
10 Dec - 17 Dec	W	7272063	7384383	7350495	7342786	29349727	81.53%
	X	13976	9801	10232	8802	42811	—
	Y	4875	3794	3932	3261	15862	—
17 Dec - 24 Dec	W	8036262	8122376	8112440	8101316	32372394	80.93%
	X	15480	11442	11926	10448	49296	—
	Y	5430	4425	4612	3825	18292	—
24 Dec - 31 Dec	W	8850633	8915370	8872494	8848908	35487405	80.65%
	X	16975	12723	12982	11573	54253	—
	Y	5939	4922	5054	4227	20142	—
31 Dec - 07 Jan	W	9660610	9694866	9647053	9608487	38611016	80.44%
	X	19092	13931	14066	12662	59751	—
	Y	6872	5411	5516	4636	22435	—

extended periods of time. The HRD system was tested using HP Pavilions with four 733MHz (20 Gigabytes of memory) Intel Celeron processors, 128 MB SDRAM, and Intel Pro/100+ Server Adapter Ethernet cards, connected via two D-Link DSH-16 10/100 dual speed hubs with switches through a 144 Kbps router. The dispatcher tests were run using Red Hat Linux release 7.0 (Guiness).

4.1.2. Web Probe Dispatchers for the HRD System

The HRD system searched the Internet for those ISPs hosting Web services for a total of twelve weeks which included 3 U.S. holidays—Thanksgiving, Christmas, and New Years. The start date was 15 October 2001 and the terminating date was 07 January 2002. Table 2 present one-week data collection periods that span from Monday to Monday. The result code legend for this table of cumulative results is

1. W — total number of probes released
2. X — total number of responding ISPs hosting HTML services
3. Y — total number of DNS name resolutions.

This table summarizes the status of each probe over 7-day collection periods. The target ratio is computed using

$$Target\ ratio = \frac{\sum_{i=0}^{3} W_i\ probes/week}{4\ \ million\ probes/week} \tag{1}$$

As evident in this table, the results reveal a steady increase in the total number of HTML servers associated with

Table 3: The ISI system input parameters.

Parameter	**Version A**	**Version B**	**Version C**	**Version D**
Dataset size	1024	1024	1024	1024
Max number of iterations	200	200	≤ 200	≤ 200
No. of hash table entries migrated new workload assignments	Unlimited	1	16	1
Rate of random NN	100%	100%	variable	variable
Rate of NNCs				
—disjoint	N/A	N/A	variable	variable
—overlapping	N/A	N/A	variable	variable
Probe set size	16 Static words	16 Static words	16 Dynamic words	16 Dynamic words
Retrieval calculations	Standard processing	Inverted (OoC) processing	Inverted (OoC) processing	Inverted (OoC) processing

each node—which are not consistent with expected results. Four dispatchers' results were examined in this test of the HRD system.

All the nodes achieved their best combined performance during the first week of each distinct study. The subsequent weeks reflect the use of the same node hosting the probe dispatcher that also hosts its accompanying scout and forager dispatchers. This reduction in node efficiency reflects the sharing of computer resources among the dispatchers. The results presented reflect offline ISP discovery and page retrieval [9] where the requirements imposed by the real-time processing delays results in each unique Web group (i.e., probes, scouts, and foragers) generating it best performance. The weekly ratios reflect subtle changes in the number of released probes, responding ISPs, DNS resolutions, and retrieved pages. These subtle changes can be seen in the weeks before Halloween, Thanksgiving, Christmas, and New Years.

4.2. ISI Results

4.2.1. ISI Execution Environments

A search engine Case Study [17] provided the motivation for the development and implementation of the initial ISI system whose goal was to demonstrate the feasibility of an EC model. A MPI cluster of nine SUN workstations was used in this study which consisted of one SUN SPARCstation 2 @ 40.0 MHz with 40 Megabytes of memory and eight SUN SPARCstation 20 (Model 61) @ 60.0 MHz with 32 Megabytes. The SUN SPARCstation 2 was treated as the Web page dispatcher ($Node_0$) and the other workstations were labeled $Node_1, \cdots, Node_8$. Web pages used in this study consisted of 1024 Yahoo Business Headline documents [19]. These studies focused on the development and testing of the stochastic regulatory mechanism—a component of the Web page dispatcher. The load balancing results were presented in Reference [18].

4.2.2. Input Parameters

Table 3 presents the ISI system input parameters for the various studies in this research effort. Versions A and B used only random NN which required 200 iterations each. Versions C and D required a reduced number of iterations since overlapping and disjoint nearest-neighbor clusters (NNCs) were allowed as supersteps [4]—multiple applications of the recombination operator.

The static probe sets for Versions A and B were created using the AWK programming language [1] and reflected the 16 most commonly occurring strings within the 1024 selected Web documents. Versions C and D relied on sets of 16 randomly chosen strings forming dynamic probe sets during each iteration in order to perform the retrieval calculations on each cluster of pages residing on a node. The dynamic probe sets are created as a component of the stochastic regulatory mechanism by randomly selecting a node, and then deriving a set of 16 search strings from its collection of words. Each version of this application was executed 80 times. Ten executions were associated with each workstation configuration that incorporated $1, 2, \cdots, 8$ indexers coupled with the page dispatcher, respectively.

4.2.3. Workload Adjustment Results

The probe sets were used as a component of the retrieval calculations associated with each indexer node, which in turn were used to create subclusters for the purpose of forming NNCs to facilitate information sharing. When performing a comparison between the computed stochastic adjustments in each node's workload assignment for these versions, the fitness measures reflect this trend. However, the probe sets associated with Version C and D were chosen randomly (for

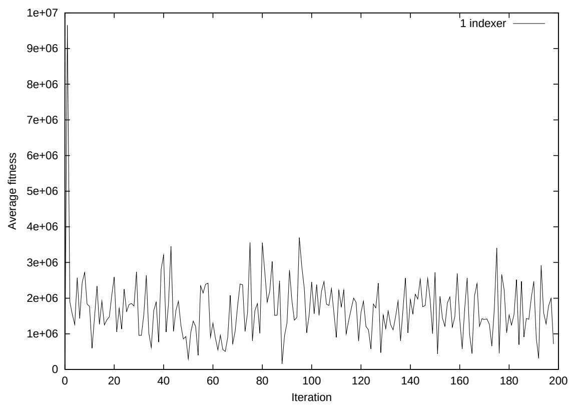

Figure 1: Version A through D—fitness measures 1 indexer (sequential).

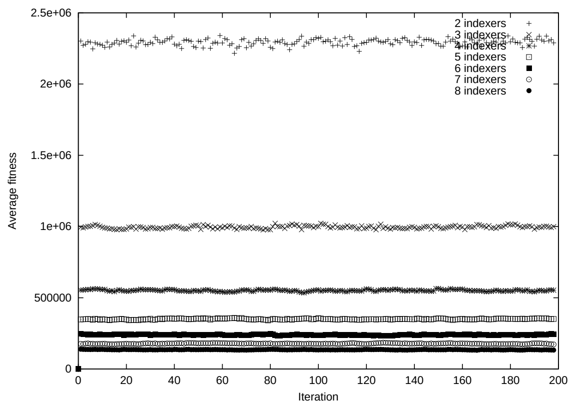

Figure 2: Version A—fitness measures for 2 through 8 indexers (Version B differs by a factor of approximately 1.0).

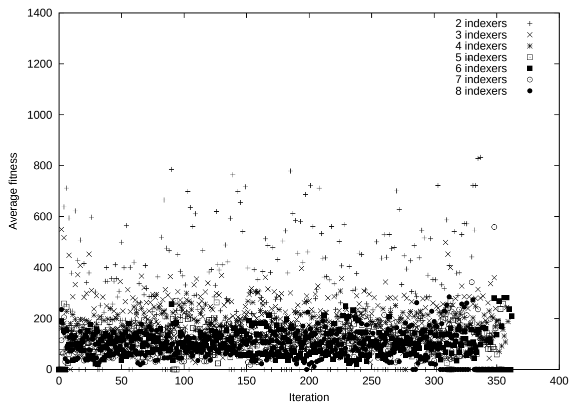

Figure 3: Version C—fitness measures for 2 through 8 indexers (Version D differs by a factor of approximately 0.571).

each iteration) which resulted in some of the fitness measures computing a best case fitness value of 0.0. This phenomenon is reflected in Figures 1 through 3. This approach was taken to convey aspects of the computed measurements that are overshadowed due to the adjustment frequencies associated with Versions C and D. The plots associated with these two versions show adequate page migration rates for all MPI clusters. Versions A and B reflect the static nature of databases associated with current search engines where Versions C and D reflect the true nature of the IS model.

The stochastic regulatory systems' parameters in Table 3 show that Versions A and B are very similar, as are Versions C and D. The differences between Versions A and B are 1) the number of migrated hash table entries, and 2) the methodology used to compute the stochastic fitness measures. Version A did not impose any size restrictions, thus resulting in enormous loads on the file server as the number of nodes were increased in the MPI cluster.

5. RELATED WORK

Internet discovery agents [13] incorporated methodologies for auto-discovery of the inter-dependencies among services provided by Internet service providers (ISPs). These agents were used to implement a management system for host supporting application servers associated with FTP, telnet, e-mail, news, DNS, NFS, Web, etc. The automated methodologies reduce the amount of human intervention needed to customize the system when new ISPs are located.

The exploration behavior of Internet agents [11, 12] for the multiple access problem (MAR) considered the search strategy of n agents accessing m servers. The retrieval of information from the various shared servers was maximized thru agent cooperation whereby each agents sends a moderate number of queries, otherwise the server's performance deteriorates leading to exponential delay as the number of queries increases.

Document centroids [14] were used to cluster static document space into distinct regions which in turn maximize the similarity between pairs of documents within a cluster. This satisfied the requirement of the average similarity between the different cluster centroids being minimized. This approach classified query terms into categories for 1) content identification, 2) components of indexing phrases, and 3) a common term (thesaurus) class.

6. SUMMARY

Stochastic optimization techniques have been used heavily in the design of Tocorime Apicu. These optimization techniques provide the foundation for improving the IS system performance by working to reduce imbalance in work loads, overlap among task workload assignments, saturation of file servers within a network (LAN, LAN+WAN, and WAN), and sensitivity to irregularities associated with Internet traffic.

This paper has related the underlying mechanisms of the Tocorime IS system to the EC model. This mapping is extended by aspects of finding hidden knowledge in a collection of documents—related and/or unrelated. Canonical Web pages were generated to reduce the workload and storage requirements of the ISI system resulting in a set of condensed documents forming the data warehouse. The ISI system continuously re-partitions the document space among a set of distributed nodes using a stochastic regulatory system whose goal is to form subclusters of nodes for redistributing the workload.

7. ACKNOWLEDGMENTS

The author wishes to thank D. Stott Parker and Gary B. Fogel for their direction and suggestions. The author wishes to express his gratitude to the reviewers whose detailed and useful comments helped tremendously to improve the quality of this paper. Also, the author wishes to thank Elias Houstis, Ahmed K. Elmagarmid, Apostolos Hadjidimos, and Ann Catlin for support, encouragement, and access to the computer systems at Purdue University. Support for this work came from Tapicu, Inc. and Honeybee Technologies.

8. REFERENCES

[1] A.V. Aho, B.W. Kernighan, and P.J. Weinberger. *The AWK Programming Language*. Addison-Wesley, Reading, MA, 1988.

[2] R. Bagrodia. Process Synchronization: Design and Performance Evaluation of Distributed Algorithms. *IEEE Transactions on Software Engineering*, 15(9):1053–1064, September 1989.

[3] T. Bắck, U. Hammel, and H. Schwefel. Evolutionary Computation: Comments on the History and Current State. *IEEE Transactions on Evolutionary Computation*, 1(1):3–17, 1997.

[4] D.C. Dracopoulos and S. Kent. Genetic Programming for Prediction and Control. *Neural Computing & Applications*, 6(4):214–28, 1997.

[5] D.B. Fogel. *Evolutionary Computation: The Fossil Record*. IEEE Press, Piscataway, NJ, 1998. Chapter 14.

[6] J.B. Free. *The Social Organization of Honeybees*. (Studies in Biology no. 81) The Camelot Press Ltd, Southampton, 1970.

[7] J.L. Gould and C.G. Gould. *The Honey Bee*. Scientific American Library, New York, 1988.

[8] K.S. Jones and P. Willett. *Readings in Information Retrieval*. Morgan Kaufmann Publishers, Inc., San Francisco, 1997. Chapter 1.

[9] T. Lizambri, F. Duran, and S. Wakid. The Best Case for TCP Management. *Journal of Network and Systems Management*, 8(4):449–456, 2000.

[10] G.E. Mobus. Foraging Search: Prototypical Intelligence. *Journal for Computing Anticipatory Systems*, 2000.

[11] J.C. Oh. Benefits of Clustering Among the Internet Search Agents Caught in the $n-$Person Prisoner' Dilemma Game. In *Proceedings of CEC 2000*, pages 864–871. IEEE, Piscataway, NJ, 2000.

[12] J.C. Oh. Cooperating Search Agents Explore More than Defecting Search Agents in the Internet Information Access. In *Proceedings of CEC 2001*, pages 1261–1268. IEEE, Piscataway, NJ, 2001.

[13] S. Ramanathan, D. Caswell, and S. Neal. Auto-Discovery Capabilities for Service Management: An ISP Case Study. *Journal of Network and Systems Management*, 8(4):457–482, 2000.

[14] G. Salton, A. Wong, and C.S. Yang. A Vector Space Model for Automatic Indexing. *Communications of the ACM*, 18:613–620, 1975.

[15] D.R. Swanson. Historical Note: Information Retrieval and the Fututre of an Illusion. *Journal of the American Society for Information Science*, 39:92–98, 1988.

[16] R.L. Walker. Preliminary Study of Web Scouts/Foragers for a Bioinformatic Application: A Parallel Approach. In Y.V. Esteve, G.M. Carlomagno, and C.A. Brebbia, editors, *Computational Methods and Experimental Measurements X*, pages 967–976. WIT Press, 2001.

[17] R.L. Walker. Search Engine Case Study: Searching the Web Using Genetic Programming and MPI. *Parallel Computing*, 27(1/2):71–89, March 2001.

[18] R.L. Walker. Using Nearest Neighbors to Discover Web Page Similarities. In H.R. Arabnia, editor, *PDPTA'02: Proceedings of the 2002 International Conference on Parallel and Distributed Processing Techniques and Applications*, pages 157–163. CSREA Press, June 2002.

[19] Yahoo. Yahoo Web Page. Yahoo Inc. Santa Clara, CA, November 1998.