Emergent System for Information Retrieval[1]

Răzvan-Dorel CIOARGĂ, Mihai V. MICEA, Bogdan CIUBOTARU,
Vladimir CREȚU, Dan CHICIUDEAN
"Politehnica" University of Timisoara
Computer & Software Engineering Department
2, Vasile Parvan Blvd., 300223 Timisoara, Romania
razvan.cioarga@cs.upt.ro, mihai.micea@cs.upt.ro,
bogdan.ciubotaru@cs.upt.ro, vladimir.cretu@cs.upt.ro,
dan.chiciudean@cs.upt.ro

Abstract. Stand alone as well as distributed web crawlers employ high performance, sophisticated algorithms which, on the other hand, require a high degree of computational power. They also use complex interprocess communication techniques (multithreading, shared memory, etc). As opposed to the distributed web crawlers, the ERRIE crawler system presented in this paper displays emergent behavior by employing extremely simple algorithms that are very efficient when dealing with a large number of entities but are unpredictable and nondeterministic. The paper discusses the ERRIE web crawler system alongside with a comparison between it and a common, stand-alone web crawling application called WIRE.

1 Introduction

The evolution of information systems and of the World Wide Web, combined with the change of perspective, from the static content to the dynamic content of the WEB 2.0 paradigm, generate an extraordinary need of storing, retrieving, indexing Internet data. In this context, extremely powerful search engines like Google, AltaVista or Yahoo use specialized software applications called web crawlers.

The web crawler is an automated software application which browses the World Wide Web in a methodical manner. The Internet search engines use web crawling as a means of providing up-to-date information regarding the queries that have been posted by their users. The web bots, as the web crawlers are often called, are usually used to make local copies of the web pages that were visited; the local copies may be processed later by the search engines. Sometimes the crawlers can be used for retrieving specific information from the Internet, such as e-mail addresses which are

[1] Supported partly by a research grant of excellence from the Romanian Ministry of Education and Research: CEEX -ET-07/2006–2008.

used for spam. Either the commercial web crawlers, or the open-source web crawlers, make use of special distributed systems algorithms requiring at least a large amount of computational power. There are some crawlers which have been specially designed to employ collaborative techniques.

Our current research focuses on developing methodologies and algorithms to solve problems related to distributed embedded systems and collaborative environments by using emergent behavior patterns.

This paper introduces a web crawler called ERRIE (Emergent Retriever of Information) that can index specific information from the World Wide Web. It is based on a custom-designed algorithm which applies emergent behavior patterns derived from the study of communication systems inside ant colonies, and of the pheromone-based communication and interaction. This bot can also be customized to retrieve information from any other electronic data storage facility.

A comparison between ERRIE and a similar, relatively commonly used web crawler called WIRE (Web Information Retrieval Environment) is also presented along with some of the most interesting results regarding their performances.

2 Web Crawling

A web crawler or a web bot is an autonomous program that downloads web pages from the World Wide Web with the purposes of: storing, backing up the pages, or indexing the web page content for later searches [1-4]. The crawler downloads the web page and parses it for valid links (either hyperlinks, like the ones contained inside the `<a>` html tag, or for other types of links, like the ones contained inside html tags such as `<script>`, `<div>` or `<img>`). Then, the bot extracts those links and repeats the process described here for all the extracted links. This algorithm is usually repeated until there are no more valid links.

A conventional web crawler has a number of problems. It is very limited by its resources, either disk space and/or network bandwidth. For all of the billion interconnected web pages a web crawler faces the problem of recurring links. Usually, the more popular the web pages, the more they are referred to by an ever growing number of other pages (this is exactly what popularity means in some concepts). As a result, the web crawler is bound to extract some links that have already been processed. A usual improvement of the conventional web crawler is to store all the visited links and compare the newly extracted links against this collection to use only the unvisited links [2]. As the number of links grows, it is important to develop faster searches (web crawlers being used by big search engines like google.com or yahoo.com, are known to handle millions of links).

Usually, a web crawler has some degree of multithreading or multiprocessing (e.g. each newly found link is placed on a queue and processed by a different thread of the crawler application until the maximum number of threads is reached). Some web bots are developed using concepts from the distributed systems field: each bot runs on a different machine or processor, each crawler handles a number of different links; the web bots use some kind of intercommunication specific to distributed systems for synchronization purposes.

Common commercial web crawlers such as the Google Crawler [5], PolyBot [6] or WebFountain [7], or common open-source web crawlers such as the GNU Wget, HTTrack or WIRE [8], face complex problems when dealing with multiprocess systems. Some of these problems have been solved by simpler, lighter collaborative crawlers, such as Ubicrawler, which do not have a central "overseer" process but make use of the multi-agent paradigm. In their case, problems specific to the multi-agent systems can be solved by applying a more natural way of system distribution by using emergent behavior patterns.

3 The WIRE Bot

The WIRE project has been created by the Center for Web Research [8] as a tool for information retrieval designed to be used on the World Wide Web. The project is currently composed of a very configurable web crawler and a simple format for storing the retrieved data, and some supplementary tools needed to extract information stored in the repository: statistics, reports etc.

The WIRE Web Crawler is an open-source library of highly scalable and configurable code (via a global XML configuration file), which also provides some analysis features for the stored information. It is composed of four main entities: the Seeder, the Harvester, the Gatherer and the Manager [8] (see Fig. 1).

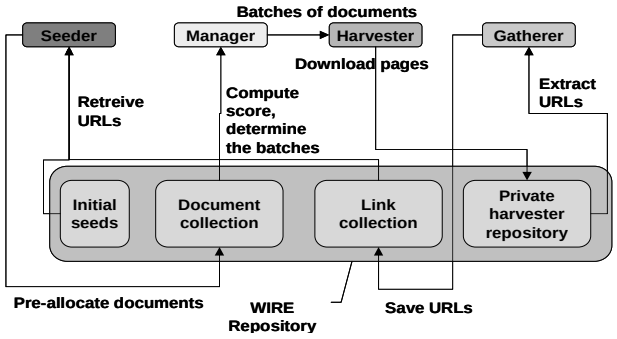

Fig. 1. WIRE system.

Each of these entities is an autonomously executed program. They communicate with each other by saving all their data in a specific folder on the hard drive, inside the WIRE repository. The entire structure of the repository is automatically created by the WIRE project; in addition, it also pre-allocates disk space for future documents. Each application saves its data inside its own specific folder; when needed, each application reads data from the other application folders.

4 eBML – Emergent Behavioral Language

Emergence is the process by which complex behavior patterns are formed starting from extremely simple rules [9]. The new field of emergent behavior is mainly based on the study of the natural world in order to retrieve simple natural algorithms that can be applied in other fields of interest. The simplicity of these patterns and the interaction between them can solve some of the most important problems in a large variety of domains, ranging from intelligent sensor networks and robotic collectives, to data mining and efficient information storage and retrieval.

The Emergent Behavioral Modeling Language or eBML [10] is loosely based on UML (Unified Modeling Language). The eBML is a formal language, allowing various constructs from Boolean guards to conditional execution, loops etc. It has a visual representation using the principles of state diagrams and a text representation. The representations are interchangeable; the text representation is mainly used for behavioral modeling while the visual representation is used as a presentation layer for the text representation. A special interpreter application is under development for the eBML. It will be able to parse a text file that contains the text representation and to generate code for a specific target. This means that a user can input the emergent behavior algorithms into a text file using eBML and then compile the text file into an executable code which can be loaded on a specific microcontroller/ microprocessor platform.

Being derived from UML, eBML presents similar formal constructs. Each eBML script starts with the keywords `behavior is` and must end with `end behavior`. Similarly, the eBML diagram must have a `START` and an `END` statement. The base construct of eBML is the activity. An activity represents an operation, a function call, a step in business process logic or an entire business process logic. Constucts such as `go(forward)` and `avoid(obstacle, direction)` are activities. Similar to UML, eBML also has some special types of activities: the "black hole" activity has transitions leading to it but has no transitions leading away from it; a "miracle" activity has only transitions leading away from it and no transitions leading to it.

Another construct of eBML is the decision point which represents points of division in the normal flow of the behavior. These decision points should reflect the previous activity as they do not have any significant label attached to them, or some condition as the flow chart decision points have. Multiple transitions with associated guards can leave from each decision point. Transitions are the constructs that appear between activities or between activities and decision points. In the visual representation of eBML, each transition is represented by an arrow which indicates the flow from one construct to another. In the text representation of eBML, transitions are not actually represented but are implied by the order of the instructions and commands.

The final basic construct of eBML is the guard which represents a condition which is associated with a transition; the transition can be traversed only when that condition is true. There are some constraints regarding decision points and guards. First, each transition that starts from a decision point must have a guard associated with it. All the guards associated with transitions that leave from a decision point must not overlap (the guards $T \leq 1$ and $T \geq 1$ overlap and there is a confusion as to

which transition should be traversed when $T = 1$). All the guards starting from a decision point must form a complete set over the possible results (the guards $T < 1$ and $T > 1$ do not form a complete set and there is a confusion regarding which transition should be traversed when $T = 1$, if any). Some transitions do not have an associated guard. These transitions are always traversed as the absence of an associated guard is equivalent to the presence of the "true" guard.

5 ERRIE (Emergent Retriever of Information)

As opposed to the existing distributed web crawlers, which require high computational power and use complex interprocess communication techniques, the ERRIE crawler system has been designed based on emergent behavior concepts by employing extremely simple algorithms that are very efficient when dealing with large number of entities. ERRIE implements a communication algorithm based on the concept of pheromones from the ant colonies.

The pheromone exchange is one of the most used ways of communication in emergent natural systems. Ant colonies are a very important example of a decentralized system that has a pheromone based communication [11]. Individual ants have a local image of the whole system, based on the pheromone exchange between them and the ants they meet. By extracting information from the pheromones, the ants determine the behavior of the neighboring ants and thus determine their own role without establishing any dialogues or any other type of negotiation. Pheromone-based communication is semi-reliable because the chemical messages continue to be transmitted even after the source of the message ceases to exist. The loss of individual messages has no importance. This type of communication is unidirectional; the pheromone trails are created by the ants for other ants to follow. There is no duplex communication; a pair of simplex messages is used instead, but they might not be correlated as part of a conversation [11].

The ERRIE algorithm introduces the MySQL Server with a MySQL database as a support for communication. The MySQL database offers support for indexing and full-text searches that are needed when retrieving links and are useful to prevent the recurrence of the already visited links (the table that stores the data has UNIQUE columns that prevent the insertion of identical data). Every access to the database is transactional (transactions are fully supported by the MySQL server) thus enabling concurrent accesses to the stored data.

The algorithm uses the persistence layer offered by the MySQL server in a way similar to the pheromone based communication described above. Every ERRIE bot leaves a mark, similar to a pheromone trail as it saves information in the database. As in the case of pheromones, these marks are very persistent; the information is stored in the database and it is not likely to disappear from there without some external user intervention. The data from the MySQL database is of different types (saved links or saved web pages) indicating different things, just as the pheromones have in the case of ant colonies. The communication between ERRIE bots takes place only through the MySQL database and, as such, it is a simplex communication. Just like simple real life ants, each ERRIE bot has only a local, very limited view of its environment

(just the pages it has currently downloaded, that are being stored in the database). The global view of the environment is clear only to an external observer.

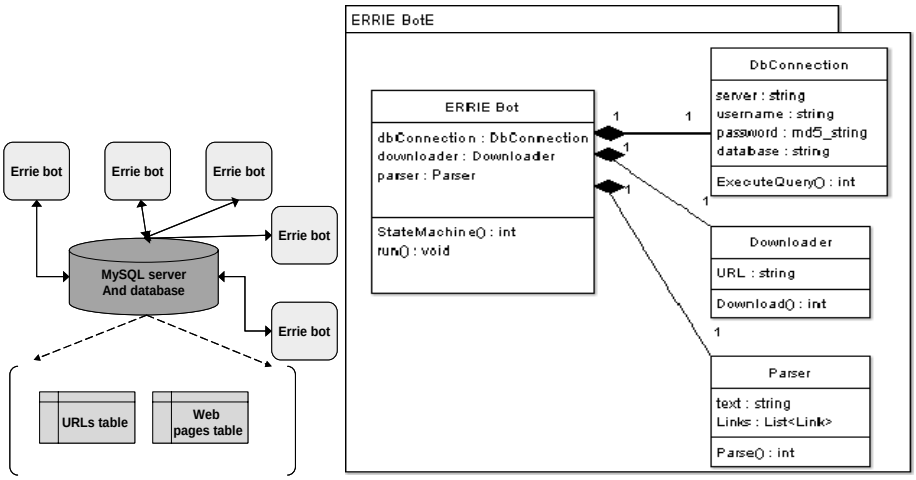

Fig. 2. ERRIE system architecture (left) and ERRIE bot class diagram (right).

The use of the open-source database server, MySQL (which provides good performance and reliability), as a communication environment and as a persistence layer, makes the intercommunication between the ERRIE bots to have similar attributes as pheromone based communication: semi-reliability, no data duplication, synchronization provided by external means (the ERRIE bots do not require special means of synchronization because this is done at the transaction level by the MySQL server). The following code presents the ERRIE bot algorithm based on the eBML language. This algorithm (*execution cycle*) describes the behavior of a single ERRIE bot as it crawls the World Wide Web.

```
behavior is
      retrieve top X unvisited links from repository
      download pages
      save pages to database
      extract URLs from pages
      save links to databse
end behavior
```

6 Experimental Results

A series of experiments have been conducted to evaluate the operation of the ERRIE system and to compare its performance to those of other web crawlers, including the WIRE bot described in a previous section.

The WIRE web crawler is extremely predictable, as it runs in cycles that are very well defined. The only amount of uncertainty has been introduced inherently by the

actual extracting of links from the web pages, as the number and type of links found in each web page is extremely unpredictable. To have similar inputs for all the experiments, the initial seeds have been set to the same value and a standard 100 KBps Internet line has been used in all the cases. The experiment conducted for the WIRE system is used as a baseline for the other experiments.

The WIRE system has been fed with initial seeds containing a large number of URLs. Then, the web crawler was started. As it runs in cycles, the number of recorded links was retrieved after each cycle (see Fig. 3, left). After approximately 24 full cycles, the WIRE system reaches a total number of 1.1 million pages downloaded in 2.21 millions seconds (more than 25 days).

On the other hand, a series of experiments have been performed to evaluate how the ERRIE system behaves on various configurations but using the same initial environment as in the case of WIRE system. Similarly to the WIRE experiment, the ERRIE system was configured to run for 24 full cycles (see the eBML code in the previous section).

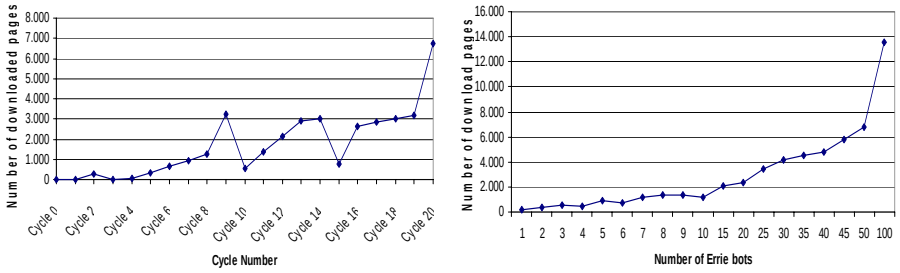

Fig. 3. The number of downloaded web pages by the WIRE System (left) and by a varying number of ERRIE bots (right).

The right half of Fig. 4 shows the results of a combination of two different experiments: both the number of ERRIE bots and the number of retrieved links are varied.

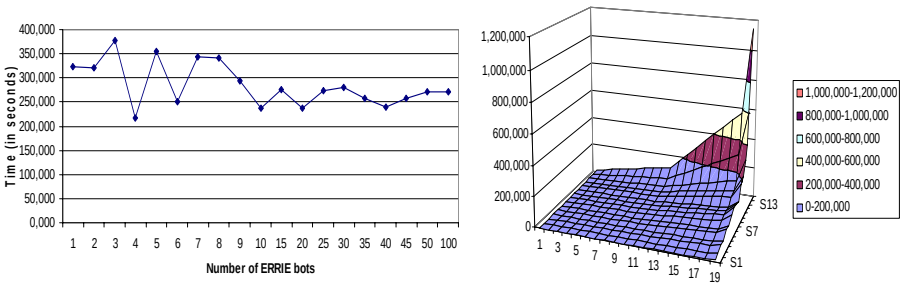

Fig. 4. The time needed to download 1.1 mil. web pages by a variable number of ERRIE bots (left), and the number of download pages (Oz) by the number of ERRIE crawlers (Ox), for a varying limit of retrieved links per crawler (Oy) (right).

The results of the experiments conducted show that the ERRIE system is more efficient when working with a relatively small number of ERRIE bots that retrieve a relatively small number of links from the MySQL database. This situation is depicted in dark grey in Fig. 4 The dark grey area in the figure shows the level of compromise in terms of the ratio of required / available network bandwidth with respect to the number of links retrieved from the database: one can either (a) browse through 100 websites by using 100 crawlers that browse 1 site each, or (b) by using 1 web crawler that browses the 100 sites, or (c) by using 10 web crawlers that browse 10 sites each. In terms of efficiency, the latter situation is the most suitable.

7 Conclusions

This paper introduces a new, emergent type of web crawler, ERRIE, which makes use of the full-text search and indexing capabilities of the MySQL database server while implementing emergent behavior principles derived from the pheromone-based communication in ant colonies. The main advantages of the ERRIE crawler are its simplicity of design and operation, as well as its efficiency.

A comparison between the emergent crawler and a common, open-source web bot has also been presented in this article, along with some of the most interesting results obtained from an extensive set of experiments which have been conducted to evaluate their performance. A compromise was made in order to make emergent web crawling more efficient, showing that when applying emergent algorithms, the most effective solution is to use middle values for both the number of crawlers and the number of links retrieved from the database.

In data retrieval applications, where multithread techniques are common, the use of emergent patterns retrieved from pheromone-based communications is very suitable, due to the efficiency of the interactions between simple entities, as compared to the resource-consuming exchanges employed in normal distributed systems.

The ERRIE system described here is an open-ended project, which can be customized to retrieve information from any other electronic data storage facility.

References

[1] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. M., K. Nigan, and S. Slattery, Learning to Construct Knowledge Bases From the World Wide Web, *Artificial Intelligence*, 118(1–2), 69–113 (2000).

[2] C. C. Aggarwal, F. Al-Garawi, and P. S. Yu, Intelligent Crawling on the World Wide Web With Arbitrary Predicates, in *Proc. 10th Intl. Conf. on World Wide Web*, 2001, pp. 96-105.

[3] S. Chakrabarti, M. van den Burg, and B. Dom, Focused Crawling: A New Approach to Topic-Specific Web Resource Discovery", in *Proc. 8th Intl. Conf. on World Wide Web*, 1999, pp. 545–562.

[4] J. Cho, Crawling the Web: Discovery and Maintenance of Large-Scale Web Data, *PhD thesis*, Stanford University, 2001.

[5] S. Brin, and L. Page, The Anatomy of a Large-Scale Hypertextual Web Search Engine, in *Proc. 7th Intl. Conf. on World Wide Web*, 1998, pp. 107-117.
[6] V. Shkapenyuk, and T. Suel, Design and Implementation of a High-Performance Distributed Web Crawler, in *Proc. 18th Intl. Conf. on Data Engineering*, 2002, pp. 357.
[7] J. Edwards, K. McCurley and J. Tomlin, Adaptive Model of Optimizing Performance of an Incremental Web Crawler, in *Proc. 10th Intl. Conf. on World Wide Web*, 2001, pp. 106-113.
[8] C. Castillo, and R. Baeza-Yates, WIRE: An Open Source Web Information Retrieval Environment, in *Wshop. on Open Source Web Information Retrieval*, 2005.
[9] V. Maniezzo, L. M. Gambardella, and F. de Luigi, Ant Colony Optimization: New Optimization Techniques in Engineering, *Springer-Verlag Berlin*, 2004, pp. 101-117.
[10] R. Cioarga, B. Ciubotaru, D. Chiciudean, M. V. Micea, V. Cretu, and V. Groza, Emergent Behavioral Modeling Language in Obstacle Avoidance, in *Proc. 24th IEEE Instrum. and Meas. Technol. Conf.*, *IMTC 2007*, Warsaw, Poland, May 2007.
[11] R. J. Anthony, Natural Inspiration for Self-Adaptive Systems, in *Proc. 15th Intl. Wshop. on Database and Expert Systems Applications*, 2004, pp. 732-736.