Computer study of the evolution of 'news foragers' on the Internet

Zsolt Palotai, Sándor Mandusitz, and András Lőrincz*

Department of Information Systems
Eötvös Loránd University
Pázmány Péter sétány 1/C Budapest, Hungary H-1117
zspalotai@vnet.hu, santi@inf.elte.hu, andras.lorincz@elte.hu

Summary. We populated a huge scale-free portion of Internet environment with news foragers. They evolved by a simple internal selective algorithm: selection concerned the memory components, being finite in size and containing the list of most promising supplies. Foragers received reward for locating not yet found news and crawled by using value estimation. Foragers were allowed to multiply if they passed a given productivity threshold. A particular property of this community is that there is no direct interaction (here, communication) amongst foragers that allowed us to study compartmentalization, assumed to be important for scalability, in a very clear form. Experiments were conducted with our novel scalable Alife architecture. These experiments had two particular features. The first feature concerned the environment: a scale-free world was studied as the space of evolutionary algorithms. The choice of this environment was due to its generality in mother nature. The other feature of the experiments concerned the fitness. Fitness was not predetermined by us, but it was implicitly determined by the unknown, unpredictable environment that *sustained* the community and by the evolution of the competitive individuals. We found that the Alife community achieved fast compartmentalization.

1 Introduction

Dynamical hierarchies of nature are in the focus of research interest [4, 23]. We have developed a novel artificial life (Alife) environment populated by intelligent individuals (agents) to detect 'breaking news' type information on a prominent and vast WWW domain. We turned to Alife to achieve efficient division of labor under minimal communication load (i.e., no direct interaction) between individuals. All components, but direct interaction, seem necessary in our algorithm to achieve efficient cooperation. There are many different aspects of this work, including evolution (for reviews on relevant evolutionary theories, see, e.g., [7, 11] and references therein), the dynamics of

* Corresponding author

self-modifying systems (see, e.g., [9, 13] and references therein), and swarm intelligence (for a review, see [14]).

A particular feature of this study is that evolution occurs in a scale-free world, the WWW [3, 15]. A graph is a scale-free network if the number of incoming (or outgoing or both) edges follows a power-law distribution ($P(k) \propto k^{-\gamma}$, where k is integer, $P(k)$ denotes the probability that a vertex has k incoming (or outgoing, or both) edges and $\gamma > 0$). The direct consequence of the scale-free property is that there are numerous URLs or sets of interlinked URLs, which have a large number of incoming links. Intelligent web crawlers can be easily trapped at the neighborhood of such junctions as it has been shown previously [16, 18]. The intriguing property of scale-free worlds is their abundance in nature [1]: Most of the processes, which are considered evolutionary, seem to develop and/or to co-occur in scale-free structures. This is also true for the Internet, the major source of information today. Thus, for the next generation of information systems, it seems mandatory to consider information gathering in scale-free environments.

The other feature of our study is that fitness is not determined by us and that fitness is implicit. Similar concepts have been studied in other evolutionary systems, where organisms compete for space and resources and cooperate through direct interaction (see, e.g., [21] and references therein.) Here also, fitness is determined by the external world *and* by the competing individuals together. Also, our agents crawl by estimating the long-term cumulated profit using reinforcement learning (for a review, see, e.g., [27]). In this estimation, a function approximation method and the so called temporal difference learning are utilized. Reinforcement learning has also been used in concurrent multi-robot learning, where robots had to learn to forage together via direct interaction [19]. The lack of explicit measure of fitness, the lack of our control over the environment, the value estimation executed by the individuals, and the *lack of direct interaction* distinguish our work from most studies using genetic, evolutionary, and reinforcement learning algorithms.

The paper is organized as follows. We review the related web crawler tools, including those [16, 18, 22] that our work is based upon, in Section 2. We describe our algorithms and the forager architecture in Section 3. This section contains the necessary algorithmic details imposed by the task, the search of the Web. Experimental results are provided in Section 4 followed by a discussion (Section 5) Conclusions are drawn in Section 6.

2 Related work

There are three main problems that have been studied in the context of crawlers. Angkawattanawit, Rungsawang [2], and Menczer [20] study topic specific crawlers. Risvik et al. [24] address research issues related to the exponential growth of the Web. Cho and Gracia-Molina [6], Menczer [20] and

Edwards et. al [10] study the problem of different refresh rates of URLs (possibly as high as hourly or as low as yearly).

An introduction to and a broad overview of topic specific crawlers are provided in [2]. They propose to learn starting URLs, topic keywords and URL ordering through consecutive crawling attempts. They show that the learning of starting URLs and the use of consecutive crawling attempts can increase the efficiency of the crawlers. The used heuristic is similar to the weblog algorithm [22], which also finds good starting URLs and periodically restarts the crawling from the newly learned ones. The main limitation of this work is that it is incapable of addressing the freshness (i.e., modification) of already visited Web pages.

Menczer [20] describes some disadvantages of current Web search engines on the dynamic Web, e.g., the low ratio of fresh or relevant documents. He proposes to complement the search engines with intelligent crawlers, or web mining agents to overcome those disadvantages. He introduces the InfoSpider architecture that uses genetic algorithm and reinforcement learning, also describes the MySpider implementation of it, which starts from the 100 top pages of AltaVista. Our weblog algorithm uses local selection for finding good starting URLs for searches, thus not depending on any search engines. Dependence on a search engine can be a suffer limitation of most existing search agents, like MySpiders. Note however, that it is an easy matter to combine the present algorithm with URLs offered by search engines.

Risvik and Michelsen [24] overview different dimensions of web dynamics and show the arising problems in a search engine model. The main part of the paper focuses on the problems that crawlers need to overcome on the dynamic Web. As a possible solution the authors propose a heterogenous crawling architecture. The main limitation of their crawling architecture is that they must divide the web to be crawled into distinct portions manually before the crawling starts. A weblog like distributed algorithm – as suggested here – may be used in that architecture to overcome this limitation.

Cho and Garcia-Molina [6] define mathematically the freshness and age of documents of search engines. They propose the Poisson process as a model for page refreshment. The authors also propose various refresh policies and study their effectiveness both theoretically and on real data. They present the optimal refresh policies for their freshness and age metrics under the Poisson page refresh model. The authors show that these policies are superior to others on real data, too. Although they show that in their database more than 20 percent of the documents are changed each day, they disclosed these documents from their studies. Their crawler visited the documents once each day for 5 months, thus can not measure the exact change rate of those documents. While in our work we definitely concentrate on these frequently changing documents.

3 Forager architecture

There are two different kinds of agents: the foragers and the reinforcing agent (RA). The fleet of foragers crawl the web and sends the URLs of the selected documents to the reinforcing agent. The RA determines which forager should work for the RA and how long a forager should work. The RA sends reinforcements to the foragers based on the received URLs.

We employ a fleet of foragers to study the competition among individual foragers. A forager has simple and limited capabilities, like a stack for a limited number of starting URLs and a simple, content based URL ordering. The foragers compete with each other for finding the most relevant documents. In this way they efficiently and quickly collect new relevant documents without direct interaction.

At first we present the basic algorithms, followed by the algorithms for the reinforcing agent and the foragers.

3.1 Algorithms

Our constraints on finding the minimal set of algorithms were as follows: The algorithms should (i) allow the identification of unimportant parameters, (ii) support the specialization of the individuals (the foragers), (iii) allow the joining of evolutionary learning and individual learning, (iv) minimize communication as much as possible. We shall return to these points in the discussion (Section 5).

Weblog algorithm and starting URL selection

A forager periodically restarts from a URL randomly selected from the list of starting URLs. The sequence of visited URLs between two restarts forms a path. The starting URL list is formed from the 10 first URLs of the weblog. In the weblog there are 100 URLs with their associated weblog values in descending order. The weblog value of a URL estimates the expected sum of rewards during a path after visiting that URL. The weblog update algorithm modifies the weblog before a new path is started. The weblog value of a URL already in the weblog is modified toward the sum of rewards ($sumR$) in the remaining part of the path after that URL:

$$newValue = (1 - \beta)\, oldValue + \beta\, sumR,$$

where β was set to 0.3. A new URL has the value of actual sum of rewards in the remaining part of the path. If a URL has a high weblog value it means that around that URL there are many relevant documents. Therefore it may worth it to start a search from that URL.

The weblog algorithm is a very simple version of evolutionary algorithms. Here, evolution may occur at two different levels: the list of URLs of the forager

is evolving by the reordering of the weblog. Also, a forager may multiply, and its weblog, or part of it may spread through inheritance. This way, the weblog algorithm incorporates the basic features of evolutionary algorithms. This simple form shall be satisfactory for our purposes.

Reinforcement Learning based URL ordering

A forager can modify its URL ordering based on the received reinforcements of the sent URLs. The (immediate) profit is the difference of received rewards and penalties at any given step. Immediate profit is a myopic characterization of a step to a URL. Foragers have an adaptive continuous value estimator and follow the *policy* that maximizes the expected long term cumulated profit (LTP) instead of the immediate profit. Such estimators can be easily realized in neural systems [25, 27, 28]. Policy and profit estimation are interlinked concepts: profit estimation determines the policy, whereas policy influences choices and, in turn, the expected LTP. (For a review, see [27].) Here, choices are based on the greedy LTP policy: The forager visits the URL, which belongs to the *frontier* (the list of linked but not yet visited URLs, see later) and has the highest estimated LTP.

In the particular simulation each forager has a $k(= 50)$ dimensional probabilistic term-frequency inverse document-frequency (PrTFIDF) text classifier [12], generated on a previously downloaded portion of the Geocities database. Fifty clusters were created by Boley's clustering algorithm [5] from the downloaded documents. The PrTFIDF classifiers were trained on these clusters plus an additional one, the $(k+1)^{th}$, representing general texts from the internet. The PrTFIDF outputs were non-linearly mapped to the interval [-1,+1] by a hyperbolic-tangent function.[2] The classifier was applied to reduce the texts to a small dimensional representation. The output vector of the classifier for the page of URL A is $\mathbf{state}(\mathbf{A}) = (state(A)_1, \ldots, state(A)_k)$. (The $(k+1)^{th}$ output was dismissed.) This output vector is stored for each URL.

A linear function approximator is used for LTP estimation. It encompasses k parameters, the *weight vector* $\mathbf{weight} = (weight_1, \ldots, weight_k)$. The LTP of document of URL A is estimated as the scalar product of $\mathbf{state}(\mathbf{A})$ and $\mathbf{weight}$:

$$value(A) = \sum_{i=1}^{k} weight_i \, state(A)_i.$$

During URL ordering the URL with highest LTP estimation is selected. (For more details, see, [22].)

The weight vector of each forager is tuned by temporal difference learning (TD-learning) [26, 28, 25]. Let us denote the current URL by URL_n, the

[2] For each class 0.5 was subtracted from the PrTFIDF value and it was then multiplied by 40. The resulting number underwent non-linear transformation by using the tanh function producing Yes/No-like outputs for the overwhelming majority of the inputs.

next URL to be visited by URL_{n+1}, the output of the classifier for URL_j by $\mathbf{state(URL_j)}$ and the estimated LTP of a URL URL_j by $value(URL_j) = \sum_{i=1}^{k} weight_i\, state(URL_j)_i$. Assume that leaving URL_n to URL_{n+1} the immediate profit is r_{n+1}. Our estimation is perfect if

$$value(URL_n) = value(URL_{n+1}) + r_{n+1}.$$

Future profits are typically discounted in such estimations as $value(URL_n) = \gamma value(URL_{n+1}) + r_{n+1}$, where $0 < \gamma < 1$. The error of value estimation is

$$\delta(n, n+1) = r_{n+1} + \gamma value(URL_{n+1}) - value(URL_n).$$

We used throughout the simulations $\gamma = 0.9$. For each step $URL_n \rightarrow URL_{n+1}$ the weights of the value function were tuned to decrease the error of value estimation based on the received immediate profit r_{n+1}. The $\delta(n, n+1)$ estimation error was used to correct the parameters. The i^{th} component of the weight vector, $weight_i$, was corrected by

$$\Delta weight_i = \alpha\, \delta(n, n+1)\, state(URL_n)_i$$

with $\alpha = 0.1$ and $i = 1, \ldots, k$. These modified weights would improve value estimation in stationary and observable environments (see, e.g, [27]and references therein), but were also found efficient in large Web environments [16, 22].

Document relevancy

A document or page is possibly relevant for a forager if it is not older than 24 hours and the forager has not marked it previously. These documents are selected out by the crawler and are sent to the RA for further evaluation.

Multiplication of a forager

During multiplication the weblog is randomly divided into two equal sized parts (one for the original and one for the new forager). The parameters of the URL ordering algorithm (the weight vector of the value estimation) are copied.

3.2 Reinforcing agent

A reinforcing agent controls the 'life' of foragers. It can start, stop, multiply or delete foragers. RA receives the URLs of documents selected by the foragers, and responds with reinforcements for the received URLs. The reward is 100 in arbitrary units for a relevant document and the cost is 1 (that is, reward is -1) for sending a document to the RA. A document is relevant if it is not yet seen by the reinforcing agent and it is not older than 24 hours. The reinforcing agent maintains the score of each forager working for it. Initially each forager

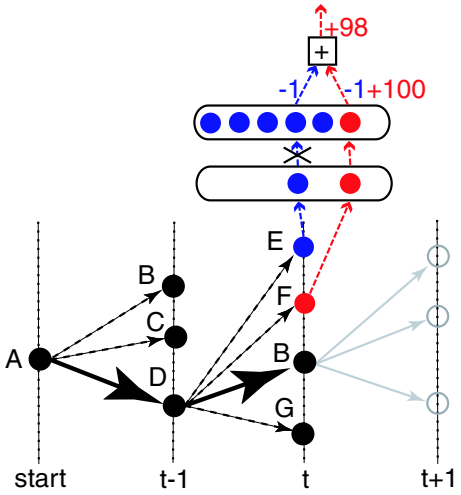


Fig. 1. Reward system
(b): Central reinforcement system. Black circles: unselected documents. Blue circles: documents that the reinforcing agent has. Red circles: novel documents. Positive (negative) numbers: reward and profit (cost). -1: negative reward for sending a document. +100: positive reward for sending a novel document. Vertical dashed lines: consecutive time steps. Dots on the $(t+k)^{th}$ dashed line: documents available at time step $t+k-1$. Solid arrows connect URLs visited by the crawler. Dashed and solid arrows point to URLs that were downloaded by the crawler at the corresponding time step.

has 100 (a.u.) score. When a forager sends a URL to the RA, the forager's score is decreased by 0.05. After each relevant page sent by the forager, the forager's score is increased by 1.

Figure 1 shows the reinforcement of a step in an example path: at start (time $t-2$), the agent is at URL 'A', where documents of neighboring URLs 'B', 'C' and 'D' are downloaded. URL 'D' is visited next. Documents of URLs 'E', 'F' and 'G' are downloaded. Document of URL 'G' has an obsolete date. Documents of URLs 'E' and 'F' are sent to the center. Document of URL 'F' is novel to the center, so it is rewarded. In turn, profit 98 is received by the forager. The forager maintains a list of neighbors of visited URLs, called *frontier*. It can only visit URLs of the *frontier*.

When the forager's score reaches 200 then the forager is multiplied. That is a new forager is created with the same algorithms as the original one has, but with slightly different parameters. When the forager's score goes below 0 and the number of foragers is larger than 2 then the forager is deleted. Note that a forager can be multiplied or deleted immediately after it has been stopped by the RA and before the next forager is activated.

Foragers on the same computer are working in time slices one after each other. Each forager works for some amount of time determined by the RA. Then the RA stops that forager and starts the next one selected by the RA.

3.3 Foragers

A forager is initialized with parameters defining the URL ordering, and either with a weblog or with a seed of URLs. After its initialization a forager crawls in search paths, that is after a given number of steps the search restarts and the steps between two restarts form a path. During each path the forager takes 100 steps, i.e., selects the next URL to be visited with a URL ordering algorithm. At the beginning of a path a URL is selected randomly from the starting URL list. This list is formed from the 10 first URLs of the weblog. The weblog contains the possibly good starting URLs with their associated weblog values in descending order. The weblog algorithm modifies the weblog and so thus the starting URL list before a new path is started. When a forager is restarted by the RA, after the RA has stopped it, the forager continues from the internal state in which it was stopped.

The URL ordering algorithm selects a URL to be the next step from the frontier URL set. The selected URL is removed from the frontier and added to the visited URL set to avoid loops. After downloading the pages, only those URLs (linked from the visited URL) are added to the frontier which are not in the visited set.

In each step the forager downloads the page of the selected URL and all of the pages linked from the page of selected URL. It sends the URLs of the possibly relevant pages to the reinforcing agent. The forager receives reinforcements on any previously sent but not yet reinforced URLs and calls the URL ordering update algorithm with the received reinforcements.

4 Experimental results

Multiple two-week long experiments were conducted for four months (between September and November). Apart from short breaks, monitoring was continuous for each two week period. Most of the figures in this article represent two weeks from November. The parameters of this experiment are representative to the entire series: The number of downloaded, sent and reinforced documents was 1,090,074, 94,226 and 7,423, respectively. The used Internet bandwidth (1Mbps on average) was almost constant, decreasing slowly by 10 % towards the end of the experiment. The number of foragers increased from 2 to 22. Experiments were run on a single computer. Foragers run sequentially in a prescribed order for approximately equal time intervals on one PC. The foraging period is the time interval during which all foragers run once. Unfinished paths are continued in the next run. Within each foraging-period, the allocated time of every forager was 180s. Some uncertainty (50 ± 30s) arose,

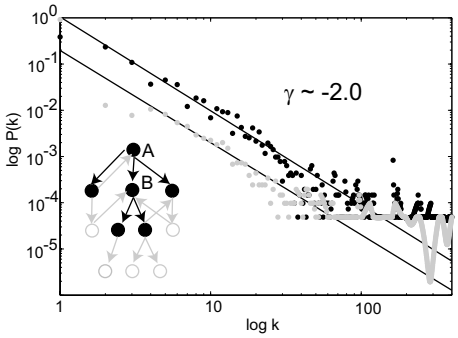


Fig. 2. Scale-free properties of the Internet domain
Log-log scale distribution of the number of (incoming and outgoing) links of all URLs found during the time course of investigation. Horizontal axis: number of edges ($\log k$). Vertical axis: relative frequency of number of edges at different URLs ($\log P(k)$). Black (gray) dots: incoming (outgoing) edges of URLs. Slope of the straight lines -2.0 ± 0.3. *Inset*: method of downloading. Black (gray) link: (not) in database. Solid (empty) circle: document (not) in database.

because foragers were always allowed to complete the analysis of the last URL after their allocated time expired. The net duration of a 100 step path was 350s.

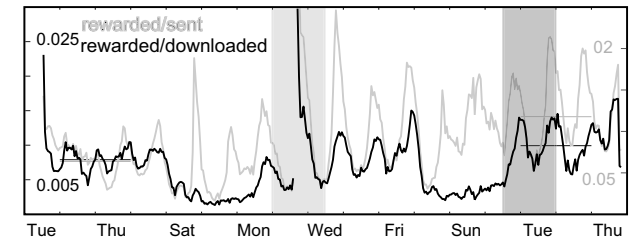
4.1 Environment

The domain of our experiments (one of the largest news sites), as most WWW domains, was scale-free according to our measurements. The distributions of both incoming and outgoing links show a power distribution (Fig. 2). The inset shows the links and documents investigated by a forager when it takes a step. The distributions, shown in the figure, correspond to links investigated by the foragers. The news forager visits URL 'A', downloads the not-yet visited part of the environment (documents of URLs, which URLs have not been visited yet by that forager and are linked from URL 'A'). Downloading is followed by a decision, URL 'B' is visited, downloading starts, and so on.

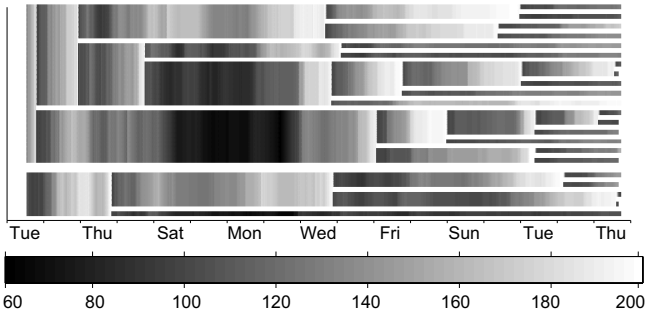
4.2 Time lag and multiplication

In the presented experiments we applied the cost/reward ratio described in Section 3.2. Larger cost/reward ratio increased the foraging areas and we noted a sudden increase in extinction probability.

Time-lag between publishing news and finding those decreases already after a few days (see Fig. 3(a)): the ratio of maxima to minima of the curves increases; and also, fewer news published on Friday were picked up on Saturday, a day late, during the second weekend of the experiment than during the



(a) Forager efficiency



(b) Population value vs. time

Fig. 3. Experimental results
(a): The rate of sent and rewarded documents showed daily oscillations. Lighter (darker) gray curve: the ratio of rewarded to sent (rewarded to downloaded) documents. Horizontal lines: average values of the two curves during two workdays. Light (darker) gray regions: number of foragers is 6 (number of foragers increases from 14 to 18). **(b):** Forager population and forager values vs. time. The scores of the foragers are given with different gray levels. Starting of white horizontal line means forager multiplication. (There was a break in our Internet connection in the middle of the second week.)

first. Further gains in downloading speed are indicated by the relative shift between lighter gray and darker gray peaks. The darker gray peaks keep their maxima at around midnight GMT, lighter gray peaks shift to earlier times by about 6 hours. The shift is due to changes in the number of sent documents. The minima of this number shifts to around 6:00 P.M. GMT. (Identical dates can be found for a 48 hour period centered around noon GMT.) Maxima of the relevant documents are at around 11:00 P.M. GMT (around 6:00 P.M. EST of the US). During the first week, the horizontal lines (average values of the corresponding curves during 2 workdays) are very close to each other. Both averages increase for the second week. The ratio of sent to reinforced documents increases more. At the beginning of the darker gray region, the relative shift of the two curves is large, at the end it is small, but becomes large again after that region, when multiplication slows down.

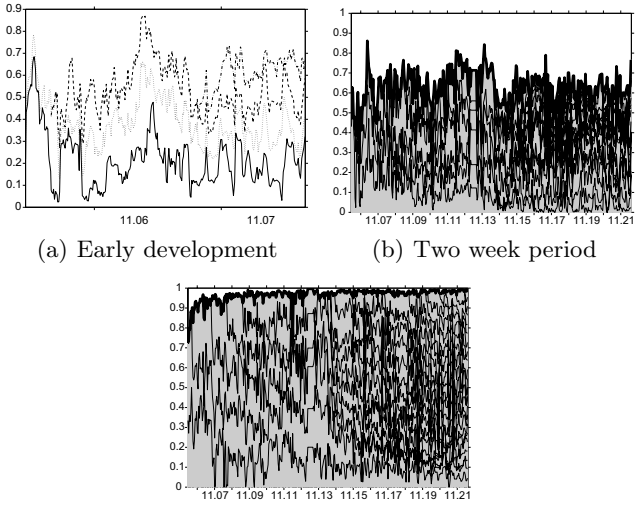
The multiplication of the foragers is shown in Fig. 3(b). Gray levels of this figure represent the value of the foragers, the range goes from 60 (darker) to 200 (lighter). In the time region studied, the values of the foragers have never fallen below 60. Upon bipartition new individuals are separated by horizontal white lines in the figure.

4.3 Compartmentalization

Division of work is illustrated by Fig. 4. Figure 4(a) show the early development of two then three and finally for four foragers. According to Fig. 4(b) large proportion of the sites are visited exclusively by not more than one forager. Only about 40% of the URLs are visited by more than one forager. Figure 4(a) demonstrates that new foragers occupy their territories quickly. Figure 4(b) shows that similar data were found for few (2-4) and for many (22) foragers (upper boundary is mainly between 0.6 and 0.7 throughout the figure). The figures depict the contributions of individual foragers: as new foragers start they quickly find good terrains while the older ones still keep their good territories. The environment changes very quickly, cca. 1200 new URLs were found every day. Time intervals of about 75 mins were investigated.

Figure 4(c) depicts the lack of overlap amongst forager trajectories. A relatively large number of sites were visited by different foragers. The question is if these foragers were collecting similar or different news. If they had the same 'goal' then they presumably made similar decisions and, in turn, their next step was the same. According to Figure 4(c) the ratio of such 2 step trajectories – i.e., the difference between the upper cover of the curves and value 1.0 – drops quickly at the beginning of the experiment, it remains very small and it decreases further as the number of foragers is growing. Given that the increase of the number of foragers gave rise to the decrease of individual foraging time, the actual numbers are only indicative. Nevertheless, the fast decrease at the beginning and the small ratio for few as well as for many foragers provides support that foragers followed different paths, that is, foragers developed different behaviors. Differences between hunting/foraging territories and/or differences between consumed food are called compartmentalization (sometimes called niche formation) [7, 9, 11, 13]. Figure 4 demonstrates that compartmentalization is fast and efficient in our algorithm.

The population of news foragers can be viewed as a rapidly self-assembling and adapting breaking news detector. The efficiency and speed of novelty detection is increasing while the structure of the environment is changing very quickly as it is shown in Fig. 5: The number of newly discovered URLs increases about linearly by time and new starting points keep emerging continuously. These drastic changes are followed by the breaking news detector, which continuously reassembles itself and maintains its monitoring efficiency. In fact, the efficiency of the detector increases with time; the quality of the assembly is improving.



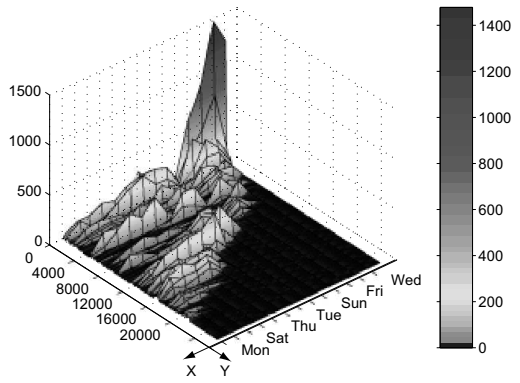
(a) Early development (b) Two week period

(c) Two step trajectories

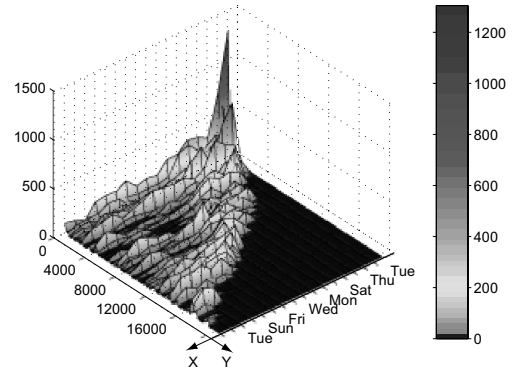
Fig. 4. Division of work
Horizontal axis in 'month.day' units. **(a):** Number of sites visited by only one forager relative to the number of all visited sites in a finite time period ($\approx 75mins$). Contribution of a single forager is superimposed on cumulated contributions of older foragers. The difference between 1.0 and the upper boundary of the curves corresponds to the ratio of sites visited by more than one forager. Duration: about three days. **(b):** Same for 16 day period. **(c):** The ratio of different two step trajectories relative to all two step trajectories conditioned that the two step trajectories start from the same site, belong to different foragers and are in a finite time period ($\approx 75mins$). Contribution of a single forager is superimposed on cumulated contributions of older foragers. The difference between 1.0 and the upper boundary of the curves corresponds to the ratio of the conditioned 2 step trajectories taken by more than one forager.

5 Discussion

We have introduced an evolutionary Alife community comprising the following main features:
Feature 1. Individuals of the community could multiply. Multiplication was kept as simple as possible. It was a bipartition process that randomly shared the properties of the parent to the descendants.
Feature 2. Individuals had two adaptive memory components: (2.a): a discrete component, the list of good foraging places, subject to evaluation and selection and (2.b): a continuous component, the behavior forming value estimating module.
Feature 3. No fitness value is available, fitness is implicitly determined by the evolution of the community and by the environment, which was not under our control.



(a) September



(b) November

Fig. 5. Distribution of new starting points
Newly discovered URLs were indexed. X axis: time in days, Y axis: index of URLs. Vertical axis: number of *new* starting points summed over equal time domains and equal domains of URL indices. As time goes on, the forager community continuously discovers new regions. Runs are from September **(a)** and from November **(b)**

Feature 4. There is a central agent, that administers the immediate reward and cost.

Feature 2.a is necessary for scale free small worlds[3]. In such structures there is a large number of highly clustered regions. In these regions, the escape

[3] Note that the distinction between scale free worlds and scale free small worlds is delicate [8]. The Internet is considered scale free small world [3, 15].

probability is very small. Intelligent crawlers get easily trapped in these regions (Fig. 6). This point has been studied thoroughly in our previous works [16, 18]. In fact, the well known PageRank algorithm had to introduce stochastic restarts to avoid the oversampling of these regions [17]. In general, feature 2.a is useful only if the forager 'knows' how to get to the foraging place. Here, this was a simple matter of memorizing the http addresses. Feature 2.a is crucial for efficient sharing of space.

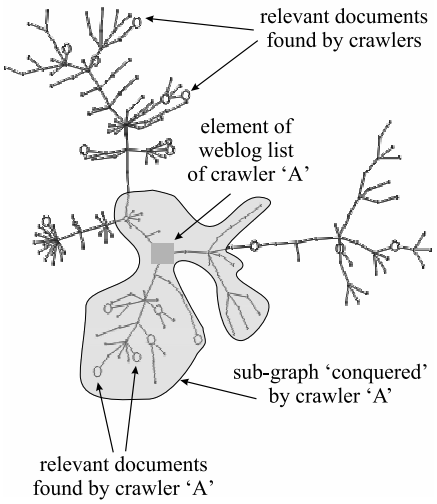


Fig. 6. Territory of crawler 'A' using one of the elements of its weblog list. Without other elements in the list the crawler is typically trapped in this domain [16, 18].

A typical path in the territory of a crawler is illustrated in Fig. 6. Without Feature 2.b, agents could divide the world into parts like the one on the illustration, but may not develop different behaviors. Feature 2.a and 2.b, together, enable different individuals to develop different function approximators and to behave differently even if they are launched from the same html page. This property has been demonstrated in Fig. 4(c). That is individuals either live in different regions (i.e., have different weblogs), or behave differently (i.e., have different function approximators), or both. This is that we call *compartmentalization* in the context of our work: each individual has its environmental niche, where the word environment concerns the environment of the Alife community. However, these niches are apparent: those are de-

termined jointly by the indirect competition, by the actual properties of the foragers and by the actual properties of the environment.

Our results indicate that agents with different behaviors have evolutionary advantages in the fast changing environment that we studied. Work sharing in the Alife community is efficient: foragers found new Internet pages continuously on the vast Internet news domain that we studied. Probably, in an environment, which is changing slowly and where food is hard to come by, the division of space and expertise in searching the owned environments could be the winning strategy. In our model, such strategy is possible. Regions of the Internet, poor in novel information, are currently under study to support this claim.

The population of news foragers can be viewed as a rapidly self-assembling and adapting news detector. The efficiency and speed of novelty detection is increasing. This occurs in spite of the fact that the structure of the environment is changing very quickly: the number of newly discovered URLs was about constant versus time. Such drastic changes are followed by the news detector, which continuously reassembles itself and improves its monitoring efficiency.

From the point of view of hierarchical organizations [4, 23], on the bottom side, the algorithm can be seen as a symbiotic collaboration between two components: the weblog is good in escaping traps, whereas reinforcement learning is good in searching neighborhoods. Such symbiotic pairs may multiply and adapt together in our algorithm. Adaptation produces a dynamic self-assembling system, which becomes closed by including the central reward administering agent into it. This closed architecture is a unit that could form a structured module at a higher level. Also, this dynamic hierarchy can be continued 'downwards': the structural modules, i.e, the selected list and the value estimator may be built from sub-components.

Finally we note that the distribution of work *without* direct interaction makes this model a possible starting point to study the emergence of interaction; the emergence of communication in our case. The intriguing point is that communication of all available information is impossible, communication of relevant information will scale badly by the number of participants if it is broadcasted, processing of communicated information consumes time, and all of these can be costly. On the other hand, properly selected and properly compressed information, which are verifiable at low risk and are robust against malicious attacks, and which can emerge and evolve under evolutionary pressure are of clear relevance in our distributed computational world.

6 Conclusions

An Alife algorithm has been introduced having individuals with two component memory. There is a discrete memory component providing a crude

'discretization' of the environment. This list develops together with the long-term cumulated value estimation memory component. Two component memory make individuals unique and efficient. Individuals behave differently, they move along different paths. Still, the efficiency of work-sharing is high and increases by time. This compartmentalization is appealing from the point of view of scalability of evolutionary systems.

The algorithm contributes to the understanding of memory components in evolutionary systems and hierarchies. Our results concern scale free small worlds, abundant in mother nature. Given that fitness is not provided by us, we see no straightforward way to show if these memory components are useful in grid worlds or not. It is possible that the speed and the efficiency of work sharing is mostly due to the highly clustered scale-free small world structure of the environment. We note that evolutionary processes seem to develop and/or to co-occur in scale-free structures. It is also true for the Internet, the major source of information today. Thus, for the next generation of information systems, it seems mandatory to consider information gathering in scale-free environments.

The algorithmic model is minimal. Many other algorithmic components, most importantly e.g., direct interaction could be included. Our point is that individuals of the community compete with each other in an *indirect* fashion and thus competition gives rise to work sharing, which looks like collaboration. Such apparent collaboration needs to be separated when the advantages of direct interaction are to be considered. It seems to us that our algorithm is a good starting point to study the advantages and the evolution of interaction by *adding* new features to the agents, which enable the development of such skills.

Acknowledgments

This material is based upon work supported by the European Office of Aerospace Research and Development, Air Force Office of Scientific Research, Air Force Research Laboratory, under Contract No. FA8655-03-1-3036. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the European Office of Aerospace Research and Development, Air Force Office of Scientific Research, Air Force Research Laboratory.

References

1. R. Albert and A.L. Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74:47–91, 2002.
2. N. Angkawattanawit and A. Rungsawang. Learnable topic-specific web crawler. In A. Abraham, J. Ruiz-del-Solar, and M. Köppen, editors, *Hybrid Intelligent Systems*, pages 573–582. IOS Press, 2002.

3. A.L. Barabási, R. Albert, and H. Jeong. Scale-free characteristics of random networks: The topology of the world wide web. *Physica A*, 281:69–77, 2000.
4. M. Bedau, J. McCaskill, N. Packard, S. Rasmussen, C. Adami, D. Green, T. Ikegami, K. Kaneko, and T. Ray. Open problems in artificial life. *Artificial Life*, 6:363–376, 2000.
5. D.L. Boley. Principal direction division partitioning. *Data Mining and Knowledge Discovery*, 2:325–244, 1998.
6. J. Cho and H. Garcia-Molina. Effective page refresh policies for web crawlers. *ACM Transactions on Database Systems*, 28(4):390–426, 2003.
7. C.W. Clark and M. Mangel. *Dynamic State Variable Models in Ecology: Methods and Applications.* Oxford University Press, Oxford UK, 2000.
8. P. Crucitti, V. Latora, M. Marchiori, and A. Rapisarda. Efficiency of scale-free networks: Error and attack tolerance. *Physica A*, 320:622–642, 2003.
9. V. Csányi. *Evolutionary Systems and Society: A General Theory of Life, Mind, and Culture.* Duke University Press, Durham, NC, 1989.
10. J. Edwards, K. McCurley, and J. Tomlin. An adaptive model for optimizing performance of an incremental web crawler. In *Proceedings of the tenth international conference on World Wide Web*, pages 106–113, 2001.
11. J.M. Fryxell and P. Lundberg. *Individual Behavior and Community Dynamics.* Chapman and Hall, London, 1998.
12. T. Joachims. A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In Douglas H. Fisher, editor, *Proceedings of ICML-97, 14th International Conference on Machine Learning*, pages 143–151, Nashville, US, 1997. Morgan Kaufmann Publishers, San Francisco, US.
13. G. Kampis. *Self-modifying Systems in Biology and Cognitive Science: A New Framework for Dynamics, Information and Complexity.* Pergamon, Oxford UK, 1991.
14. J. Kennedy, R.C. Eberhart, and Y. Shi. *Swarm Intelligence.* Morgan Kaufmann, San Francisco, USA, 2001.
15. J. Kleinberg and S. Lawrence. The structure of the web. *Science*, 294:1849–1850, 2001.
16. I. Kókai and A. Lőrincz. Fast adapting value estimation based hybrid architecture for searching the world-wide web. *Applied Soft Computing*, 2:11–23, 2002.
17. R. Lempel and S. Moran. The stochastic approach for link-structure analysis (salsa) and the tkc effect. *Computer Networks*, 33, 2000.
18. A. Lőrincz, I. Kókai, and A. Meretei. Intelligent high-performance crawlers used to reveal topic-specific structure of the WWW. *Int. J. Founds. Comp. Sci.*, 13:477–495, 2002.
19. M.J. Mataric. Reinforcement learning in the multi-robot domain. *Autonomous Robots*, 4(1):73–83, 1997.
20. F. Menczer. Complementing search engines with online web mining agents. *Decision Support Systems*, 35:195–212, 2003.
21. E. Pachepsky, T. Taylor, and S. Jones. Mutualism promotes diversity and stability in a simple artificial ecosystem. *Artificial Life*, 8(1):5–24, 2002.
22. Zs. Palotai, B. Gábor, and A. Lőrincz. Adaptive highlighting of links to assist surfing on the internet. *Int. J. of Information Technology and Decision Making*, 4:117–139, 2005.
23. S. Rasmussen, N.A. Baas, B. Mayer, M. Nilsson, and M.W. Olesen. Ansatz for dynamical hierarchies. *Artificial Life*, 7(4):329–354, 2001.

24. K. M. Risvik and R. Michelsen. Search engines and web dynamics. *Computer Networks*, 32:289–302, 2002.
25. W. Schultz. Multiple reward systems in the brain. *Nature Review of Neuroscience*, 1:199–207, 2000.
26. R. Sutton. Learning to predict by the method of temporal differences. *Machine Learning*, 3:9–44, 1988.
27. R. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, 1998.
28. I. Szita and A. Lőrincz. Kalman filter control embedded into the reinforcement learning framework. *Neural Computation*, 2003. (in press).