IJICTR International Journal of Information & Communication Technology Research

Volume 4- Number 4- September 2012(33-42)

IECA: Intelligent Effective Crawling Algorithm for Web Pages

Mohammad Amin Golshani
Department of Electrical and Computer Engineering,
Yazd University
Yazd, Iran
Golshani.ma@yahoo.com

AliMohammad ZarehBidoki
Department of Electrical and Computer Engineering,
Yazd University
Yazd, Iran
AliZareh@Yazduni.ac.ir

Received: April 5, 2012- Accepted: August 27, 2012

Abstract— **Obtaining important pages rapidly can be very useful when a crawler cannot visit the entire Web in a reasonable amount of time. Several Crawling algorithms such as Partial PageRank, Batch PageRank, OPIC, and FICA have been proposed, but they have high time complexity or low throughput. To overcome these problems, we propose a new crawling algorithm called IECA which is easy to implement with low time O(E*logV) and memory complexity O(V) -*V* and *E* are the number of nodes and edges in the Web graph, respectively. Unlike the mentioned algorithms, IECA traverses the Web graph only once and the importance of the Web pages is determined based on the logarithmic distance and weight of the incoming links. To evaluate IECA, we use three different Web graphs such as the UK-2005, Web graph of university of California, Berkeley-2008, and Iran-2010. Experimental results show that our algorithm outperforms other crawling algorithms in discovering highly important pages.**

Keywords- search engines; Web crawling; Web graph; logarithmic distance; reinforcement learning; World Wide Web.

I. INTRODUCTION

One of the most challenging issues for Web search engines is finding high quality Web pages for users. To make the Web more interesting and productive, we need efficient ranking algorithms for crawling and searching. Gulli and Signorini [1] have shown that search engines do not index the entire Web. Therefore, the focus should be on the most valuable and appealing pages. To do this a better crawling technique is required and a more efficient mechanism has to be applied. This enables search engines to present the most important and relevant pages to the user in response to her query.

The crawler is one of the main components of the search engines. It is a program for the bulk downloading of the Web pages [2]. Crawlers are given a starting set of Web pages (seed pages) as their input, extract outgoing links appearing in the seed pages and determine what links to visit next based on certain criteria. Web pages pointed to by these links are downloaded and stored in a local repository. Crawlers continue visiting the Web pages until a desired number of the pages have been downloaded or until local resources (such as memory and storage) are exhausted [3, 4]. Over the time frame of crawler technology development, the Web has been growing rapidly, and so crawlers need to operate efficiently and effectively. Most of the crawlers will not be able to visit every page for three main reasons:

- The network bandwidth is expensive [5].
- The crawlers may have limited storage capacity, so it is reasonable to expect most of the crawlers will not be able to cope with all data [6].

- Crawling takes time, so at some points the crawler should revisit previously scanned pages to prevent index inconsistency [6].

The Crawling algorithms usually use a ranking mechanism to calculate the importance of pages as a crawling priority. In other words, a ranking algorithm is applied to the visited Web graph and pages with higher ranks will have higher priority for crawling [7]. The design of a good crawler presents many challenges. The crawler must avoid overloading Web sites or network links. The crawler must deal with huge volumes of data. Unless it has unlimited computing resources and unlimited time, it must carefully decide what URLs to scan and in what order. The crawler must also decide how frequently to revisit pages it has already seen, in order to keep its client (e.g. search engines) informed of changes on the Web.

In this paper, we address one of these important challenges: How should a crawler select URLs to scan from its queue of known URLs? We propose a new method, called IECA (Intelligent Effective Crawling Algorithm), which has low time and memory complexity, and higher performance than the former algorithms. It acts based on the links between Web pages.

The remainder of this paper is structured as follows: The next section reviews the background and related work. In Section III, we introduce our algorithm, IECA. Experimental analysis and comparison to some of the well-known algorithms are given in Section IV. Section V and VI discuss complexity issues of IECA and the major contributions of the paper, respectively, and finally our conclusion and future work of research are presented in section VII.

II. RELATED WORK

Web crawlers have been studied since the advent of the Web. Nowadays, crawling algorithms are the subject of extensive researches. These studies can be categorized into one of the following topics [6]: Crawler architecture, page selection, page update (freshness), and change frequency estimation for the Web pages. This paper is placed into the page selection category.

By retrieving important pages earlier, a crawler can improve the quality of the downloaded pages. Methods based on link analysis have been widely used to calculate the page importance such as HITS [8] and PageRank [9]. In the following some of the well-known algorithms are considered.

PageRank is a popular ranking algorithm used by Google to measure the importance of the Web pages. PageRank weights each link based on the importance of the document from which it originates and the number of outlinks in the origin document. It models the users' browsing behaviours as a random surfer model [10, 11]. In this model a person surfs the Web by randomly clicking links on the visited pages. When she (PageRank) reaches to a Web page that does not have any outward link, she will randomly jump to another page. PageRank assumes that a user either follows a link from the current page or jumps to a random page on the Web graph. The rank of page j is then computed by the following equation:

$$r(j) = \frac{1-d}{n} + d * \sum_{i \in B(j)} r(i)/o(i) \qquad (1)$$

where n is the number of the Web pages, $O(i)$ denotes the number of outgoing links from page i and $B(j)$ shows the set of pages that point to page j. Parameter d, damping factor, is used to guarantee the convergence of PageRank and remove the effects of sink pages (pages with no outputs).

There is a similar work in which a new metric called RankMass has been proposed to find highly important pages [12]. The RankMass metric is based on commonly used variations of PageRank such as Personalized PageRank [13] and TrustRank [14] which assumes that users' random jumps are limited to a set of specified pages.

Najork and Wiener used the Breadth-first algorithm as a crawling algorithm. They examined the average quality of downloaded pages during a Web crawling and connectivity-based metric PageRank was used to measure the quality of the downloaded pages [4].

Cho, Garcia-Molina, and Page compared some crawling algorithms including PageRank, Backlink count, and Breadth-first [15]. It was found that the crawling based on PageRank finds the hot (important) pages earlier than others.

Abiteboul, Preda, and Cobena [16] proposed an algorithm called OPIC, to find the importance of Web pages online in the crawling process. In their method, each page has a value called cash. Initially all pages have the same cash equal to 1/n (n is the number of Web pages). The crawler will download Web pages with higher cash and when a page is downloaded, its cash will be distributed among the pages it points to. In this method, each page will be downloaded many times leading to increasing crawling time. Unfortunately, the experiments were done on a synthetic Web graph including at most 600,000 nodes with the power law distribution.

A site-based method named largest site first has been proposed [17]. In this method the sites with the larger number of pending pages have higher priority for crawling. It was found that this algorithm is better than the Breadth-first method.

A crawling algorithm has been proposed to schedule Web pages for (re)downloading into a search engine repository [11]. The objective of the algorithm is to maintain the freshness of the search engine's index based on a quality metric using users' experiences.

Dasgupta, Ghosh and Kumar et al. [18] proposed a new crawling algorithm in order to discover newly-arrived content on the Web. They measured the overhead of discovering new content, defined as the average number of fetches required to discover a new page. They showed that with perfect foreknowledge of where to explore for links to new content, it is possible

to discover 90% of all new content with under 3% overhead and 100% of new content with 9% overhead.

ZarehBidoki, Yazdani, and Ghodsnia proposed an intelligent crawling algorithm based on Q-learning, called FICA [7] which modelled random surfer user. In FICA, the priority for crawling pages is based on the concept of punishment. In their method, the aim is to minimize sum of received punishments by the Web pages so that a page with the lowest punishment will have the highest priority for crawling.

ZarehBidoki and Yazdani evaluated FICA as a ranking algorithm [19], the results show using FICA as a ranking algorithm is acceptable.

Dikaiakos, Stassopoulou, and Papageorgiou presented a study of crawler behaviour based on Web-server access logs from five different sites in three countries [20]. Their logs capture the HTTP traffic of these sites for periods ranging from 42 days to 6 months at the beginning of year 2002. Based on the logs, they analyzed the activity of different crawlers that belong to four major, general-purpose search engines (Google, AltaVista, Inktomi, and FastSearch) and one major Digital Library and search engine for scientific literature (CiteSeer). Their results and observations provide useful insights into crawler.

Baeza-Yates and Castillo [21] observed that although the Web graph is effectively infinite, most user browsing activity is concentrated within a small distance of the root page of each Web site. Hence, a crawler should concentrate its activities there, and avoid exploring too deeply into any one site. They proposed several probabilistic models for user browsing in infinite Web site. The proposed models are validated against real data on page views in several Web sites, showing that, a crawler needs to download just a few levels, no more than 3 to 5 clicks away from start page, to reach 90% of the pages that users actually visit.

III. IECA

IECA crawling algorithm model has two important features:

- IECA uses three elements of the Web graph in the importance computations of the Web pages, out-degree of parent nodes, in-degree of child nodes, and the Web structure feature (it is introduced in the next subsection).
- It uses the entire input links in calculations. When a child node has different parents in the Web graph, all of the discovered parents participate in the calculations.

In IECA, we used two introduced metric in FICA [7] and Average-clicks [22], the first is the link weight and the second is the logarithmic distance:

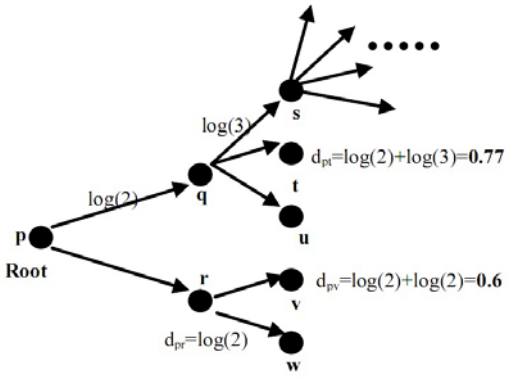

Figure 1. Logarithmic distance in the crawling tree.

Definition 1. Link weight: if page *i* points to page *j* then the weight of the link from *i* to *j* is equal to $\log_{10}O(i)$ where O(i) denotes *i*'s out-degree.

Definition 2. Logarithmic distance: the distance between pages *i* and *j* is the weight of the shortest path (the path with the minimum value) or sum of the link weights in the shortest path from *i* to *j* which is denoted with d_{ij}. Moreover, the logarithmic distance between the root (starting points in the crawling process, seed pages) and page *i* is denoted with d_i.

For example, in Fig. 1 the weight of outward links in pages *p*, *q*, and *s* are equal to log2, log3, and log4, respectively. The distance between *p* and *t* is log2+log3 and between *p* and *v* is log2+log2. Thus, whereas both *t* and *v* are the same number of links away from *p* (two clicks), *v* is closer to *p* in terms of logarithmic distance ($d_{pv}<d_{pt}$) and is more important than page *t*.

If a crawled page *i* has distance d_i from the root page, by using Definition 2 the distance of each of its child nodes is computed as follows:

$$d_j = \log(O(i)) + d_i \qquad (2)$$

If Eq. (2) is used as the selection criteria, after passing several iterations, the values of *log(O(i))* and d_i are not comparable (d_i would be much greater than *log(O(i))*) and almost the effect of the current link's weight will be lost. So, we propose the following formula which is similar to the reinforcement learning algorithm [23].

$$d_j = \log(O(i)) + w * d_i \qquad (3)$$
$$0.1 \le w \le 0.35$$

The distance factor w is used to regulate the effects of parent nodes. For example, if there is a path like $i \to k \to l \to j$, then the effect of distance of *i* on *j* is w^3. Eq. (4) shows the main formula of IECA which is based on reinforcement learning.

$$d_{j_{t+1}} = (1-\alpha) * d_{j_t} + \alpha * (\log(O(i)) + w * d \qquad (4)$$
$$i \in B(j),\ 0 < \alpha < 1,\ 0.1 \le w \le 0.35$$

TABLE I. COMPARISON OF SEVERAL SHOPPING WEBSITES (APRIL 2012).

WebSites	Backlink counts		Alexa Rank
	Google	Bing	
www.amazon.com	1270	219 million	11
www.overstock.com	457	10 million	653
www.homeshop18.com	35	200 thousand	2017
www.shoppersstop.com	30	16600	10609
www.iloveshopping.ie	22	532	620109
www.shopgotham.com	10	453	3.1 million

where α is the learning rate that is modelled in Eq. (5) and *log(O(i))* is the link weight from *i* to *j*. The old distances, d_{j_t} and d_{i_t}, show the distance values of pages *j* and *i* in time *t* respectively, and $d_{j_{t+1}}$ is the new distance of page *j* at time *t+1*.

$$\alpha = e^{-\beta * t} \tag{5}$$

In Eq. (5) *t* shows time and $\beta = 0.1$ is a static value to control learning rate, α.

Initially the crawler has little knowledge about the Web pages (environment), hence $\alpha = 1$ and selects pages based on the current status. As it visits more pages, it slowly learns from environment (α decreases). Because of little knowledge in the initial stages of the crawling process, wrong choices are undeniable. In the next section we introduced an interesting feature of the Web structure that solved the problem.

A. An Interesting Feature of the Web Graph

To better clarify our idea we would introduce two rules:

Rule 1. There is a simple rule in the Web graph. The more important a web page is, the more backlinks it should have. To check validity of the rule 1 several shopping Web sites are compared in table I.

The Breadth-first crawling algorithm traverses a Web graph by following its links. The distance of each crawled page from the root (seed pages) is always less or equal than of the uncrawled pages. The Breadth-first ordering is not the best method for the crawling [15], but it can discover pages with high PageRank in the initial stages of the crawling process. Important pages have many backlinks from different Websites, so in the Breadth-first algorithm high-quality pages have more chance to be find earlier (rule 1), Najork and Wiener experiments show our suppositions is true [4]. In their Experiments PageRank was used as the benchmark. Fig. 2 shows the average PageRank (unnormalized) of all downloaded pages on each day of the crawl.

The average score of crawled pages on the first day is 7.04, more than three times the average score of 2.07 for crawled pages on the second day. The average score decreases to 1.08 on the first week, then to 0.84 after the second week, and 0.59 after the fourth week.

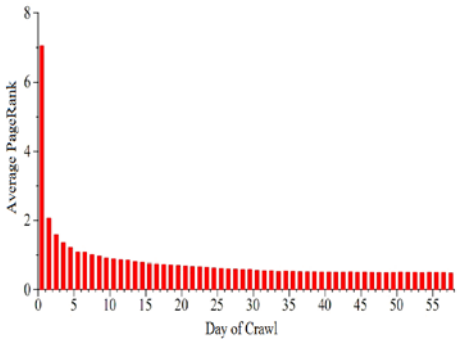

Figure 2. Average PageRank score by day of crawl [4].

IECA is based on the breadth first algorithm with a new definition of distance between web pages. Its traversal is similar to the Breadth first traversal, especially in the initial stages of the crawling process (because the Web graph is incomplete).

To use the Web structure feature (Rule 1), the first solution is merging Backlink count and IECA. But this point should not be ignored that Backlink count assumes the Web environment is flat and there is no difference between links from different Web pages with different weightings. Thus, it suffers from link farms. A link farm is a collection of artificial highly interlinked Websites created for the sole purpose of trying to hoodwink a search engine into thinking that particular Websites were more popular than they really were. As a substitute of the Backlink count method, we introduce a new definition.

Definition 3. P-C[1] coefficient: it is possible a page has several parents (several backlinks), thus to make distinction between parents, we define a function as follows:

$$f(d_{j_t}, d_{i_t}) = \frac{d_{j_t}}{d_{i_t}} \tag{6}$$

in the above formula each incoming link is weighted based on the parent node and child node distances (*j* is child of *i*). In other words, the weighting function shows how many times the child node compared to the parent node is further than the root pages. The more the distance, the more valuable the incoming link is. For example if the parent node distance is 1 and the child node distance equals 2, the weight of the incoming link is 2. Note that the parent node distance is always less than or equal to the its child node distance(s) (because we choose a Web page that has the minimum distance from queued pages).

We change Eq. (4) as the following using $f(d_{j_t}, d_{i_t})$ factor.

$$d_{j_{t+1}} = (1-\alpha) * d_{j_t} + \alpha * (\log(O(i)) + w * d_{i_t}) - \log(f(d_{j_t}, d_{i_t})) \tag{7}$$

According to Eq. (7), the distance would be reduced if the Web page has a lot of valuable in-degrees.

[1] Parent-Child.

Eq. (6) applies in situation that the signs of the parent node and child node distances are the same, but by using Eq. (7), some pages will get negative distances if they have a lot of valuable input links. However, it is impossible for the Web pages to have negative distance in Eq. (7) - the logarithm of a negative number is not defined. To fix the problem, we change Eq. (6) as the following equation (Eq. (8)).

$$f(d_{j_t}, d_{i_t}) = \begin{cases} \dfrac{d_{j_t}}{d_{i_t}} & \text{Parent node and child node have the same sign.} \\ \dfrac{d_{j_t} - d_{i_t}}{|d_{i_t}|} & \text{Otherwise.} \end{cases} \quad (8)$$

Rule 2. In this section we would introduce the second rule.We believe in different stages of the crawling process, different fraction of the parent node distance should be transfer to the child node (d_{i_t}). In the initial stages of the crawling, the discovered pages usually are more important and they should have low distances compared to the pages that are discovered in the middle or end of the crawling process. So in the initial stages of the crawling process, the amount of transferred distance (d_{i_t}) should be lower than the transferred distance in the final stages of the crawling process. Therefore, the transferred distance is considered as a variable ($(1-\delta_t)*d_{i_t}$), Eq. (9).

$$d_{j_{t+1}} = (1-\alpha)*d_{j_t} + \alpha*(\log(O(i)) + (1-\delta_t)*d_{i_t}) - \log(f(d_{j_t}, d_{i_t})) \quad (9)$$

Experimentally, we found δ_t , should be linearly decreased. We should set δ_t so that when it is close to ε , we have navigated almost the whole of the Web graph and there are not any uncrawled pages in the queue. If δ_t reaches ε too early, the throughput of the algorithm will be decreased.

At the first glance, P-C coefficient may seem redundant because if a page has a good parent (low distance), its distance will be decreased. However, in the initial stages of the crawling process, backlinks are very important, so we should make relationship between Web structure feature (rule 1) and the obtained knowledge by the crawler. Therefore we change Eq. (9) to Eq. (10).

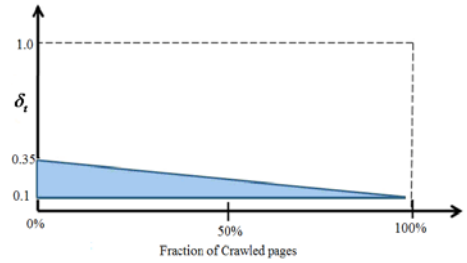

Figure 3. The relationship between δ_t and crawled Web pages

$$d_{j_{t+1}} = (1-\delta_t)*((1-\alpha)*d_{j_t} + \alpha*(\log(O(i)) + (1-\delta_t)*d_{i_t})) - \delta_t*\log(f(d_{j_t}, d_{i_t})) \quad (10)$$

In other words, we can explain Eq. (10) in another form (Eq. (11)):

$$\text{distance} = (1-\delta_t)*(\text{obtained knowledge from web environment}) - \delta_t*(\text{web structure feature}) \quad (11)$$

δ_t is balancing factor in time *t*. We use δ_t to create balance between gained knowledge by the crawler and the Web structure feature. Hence, in the early stages of the crawling process the crawler has no background about Web structure, δ_t has its maximum value. Over times as it accumulates more knowledge about the environment, δ_t will be reduced linearly. If the initial value of δ_t be in the range of [0.3, 0.35], the algorithm will have the highest throughput, and at the end of crawling process it will reach $\varepsilon \leq 0.1$. In the experiments, we modelled the balancing factor, δ_t as in Eq. (12):

$$\delta_t = -0.28*\text{Percentage of crawled Web pages} + 0.35 \quad (12)$$

The effect of the Web structure feature (δ_t) for discovering hot pages is shown in Fig. 3. Eq. (10) is used when the Web page has been previously discovered and now it is in the queue (a queue for storing the list of URLs to download and a page with minimum distance has the highest priority to be selected for crawling), so its distance will be updated by discovering each of its parents. For the new discovered pages we can not use Eq. (10), because their previous distances (d_{j_t}) are not available. To overcome this problem, we introduce a new formula for the pages that are seen for the first time in the Web graph. As was mentioned, a page that has been discovered in the initial stages of the crawling process is usually more important than a page that will be discovered at end of the crawling process. Therefore, Eq. (13) is used for calculating the distance of recently discovered pages.

$$d_{j_{t+1}} = (1-\delta_t) * d_{i_t} + \log(O(i)) \quad (13)$$

In other words:

$$d_{j_{t+1}} = (1-\delta_t) * \text{distance of parent node} + \text{the weight of the link between parent node and child node} \quad (14)$$

The pseudo code of the proposed crawling algorithm is shown below.

Algorithm 1. IECA Crawling algorithm.

```
1. d←{∞, ∞, ∞, . . .}
2. input: staring_URLs, k = 250000, t = 0,
3.        b = 0.1, size = 0, distance = 0.3,
4. enqueue(URL_queue, starting_URLs, distance);
5.  while (not empty(URL_queue))
6.              (URL, distance) = dequeue(URL_queue);
7.              size = size + 1;
8.    t = size div k;
9.   if ((size % k) == 0)
10.       δ = -0.28*percentage of crawled Web pages + 0.35;
11.             // δ is the balancing factor
12.   end if
13.       α = e^(-β * t);
14.       crawl_page(url);
15.       for each child u of url
16.          if ((d[u] != ∞) && (u ∉ crawled_pages))
17.             d_u = (1- δ)*|(1- α) * d[u] + α * (log(O(url))
                  + (1-      δ)  *  distance| – δ  *  log(f(D[u],
distance));   18.          // du is a new distance for page u
19.                      enqueue(URL_queue(u, d_u));
20.                      d[u] = d_u;
21.          else if (d[u] ==∞)      // u is a new discovered Web
page
22.                      d_u = (1- δ) * distance + log (O(url));
23.               enqueue(URL_queue(u, d_u));
24.          end if
25.      end for
26.  end while
```

The URL_Queue is a priority queue, containing pages waiting to be crawled in which the priority is the distance value. In this mechanism, the distance of each page, in addition to its URL, is inserted into the URL_queue and the pages are sorted by their distances in ascending order. *d[j]* shows the current distance value of node *j* and is set to a big value initially. The crawled_pages variable includes recently crawled pages. The crawl_page(url) function fetches a page from the Web and extracts its links. The enqueue(queue (element, d)) function inserts an element with distance *d* into its position in the priority queue. dequeue(queue) removes the element at the beginning of the priority queue along with its distance and returns them.

The number of iterations or *t* is incremented after crawling every *k* set Web pages. We found that *k* equals to 250,000 and β = 0.1 are suitable. The *variable distance* is a temporary variable, which is used to keep the distance of parent node. Starting pages should have an initial distance (when a page is discovered the distance of its parent is required). To have the minimal effect of parent node on the distances of child pages, we initialized the distances of starting pages with a small number, like 0.3. Because δ_t is less than 1, so the dependency of it on the root selection is very low. For example if page *j* is *l* links farther away than the root page (*l* clicks), then the distance effect of the root page on this page is a factor of $(1-\delta_t)^{l+1}$.

IV. Experimental Results

In this section, we report the results of our algorithm evaluation. For evaluation of IECA, we used three different Web graphs, the UK-2005 [24, 25], Web graph of Berkeley University[2], and Iran-2010[3]. Our goal was to compare IECA with other crawling algorithms to see which one of them finds more important pages (high PageRank) faster. We would compare our algorithm with the following crawling algorithms:

- Breadth-first: The crawling process is done in the breadth-first order. Initially, the algorithm starts with some starting URLs as the roots of the crawling tree.
- Backlink count: In this algorithm, pages with more input links are crawled first [15], that is, pages with more input links have higher ranks.
- Batch PageRank: This strategy calculates an estimation of Pagerank, using the pages seen so far, every K pages downloaded. The next K pages to download are the pages with the highest estimated Pagerank. This strategy was also studied by Cho et al. [9, 15], and it was found to be better than Backlink count. However, Boldi et al. [26] showed that the approximations of Pagerank using partial graphs can be very inexact.
- Partial PageRank: This is like Batch pagerank, but in between PageRank re-calculations, a temporary pagerank is assigned to new pages using the sum of the PageRank of the pages pointing to it divided by the number of out-links of those pages [27].
- OPIC: In this algorithm, all pages start with the same amount of cash [16]. Every time a page is crawled, its cash is distributed to its outward links. In each step the next page for crawling is the one with the highest amount of cash up to now.
- FICA: It is an intelligent crawling algorithm based on reinforcement learning. The priority for the crawling pages is based on the received punishments by each Web page. Pages with the lowest punishments have the highest priority for crawling.

IECA uses algorithm 1 for the crawling process. Initially, we start crawling the Web with every algorithm with some starting URLs. To reach the highest coverage on the Web graph, we select starting URLs (seeds) in the range of 5000 to 17000. Every time by crawling *k* new Web pages (*k* is set to 250,000), we run one of the above ranking algorithms. Afterward, we sort the Web pages in the queue

[2] This data set has been gathered by WIRE Crawler-2008.
[3] This dataset has been crawled by Heritrix crawler-2010.

according to the produced ranking. This process continues until a specified portion of the Web is crawled. The general crawling is depicted as algorithm 2. All methods would run in this way with their own ranking criteria [15]. Unlike other ranking algorithms, IECA, FICA, and OPIC are scheduling algorithms and they do not require any additional ranking stage.

TABLE II. ALGORITHM THROUGHPUTS IN THE DIFFERENT WEB GRAPHS.

Algorithm	Fraction of crawled hot pages			Average Throughput
	Berkeley (4 million Web pages)	UK (10 million Web pages)	Iran (3 million Web pages)	
IECA	**75%**	**57%**	**80%**	**71%**
FICA	70%	49%	65%	**61%**
Partial PageRank	73%	53%	73%	**66%**
Batch PageRank	69%	51%	72%	**64%**
OPIC	68%	51%	70%	**63%**
Backlink Count	65%	49%	74%	**62%**
Breadth-first	**65%**	**48%**	**68%**	**60%**

Algorithm 2. The general crawling algorithm.

```
1.  input : starting_urls, k = 250,000
2.  enqueue(URL_queue, starting_URLs)
3.   while (not empty(URL_queue))
4.        url = dequeue(URL_queue);
5.        crawl_page(url);
6.        for each child u of url
7.             if ((u ∉ URL_queue) && (u ∉ crawled_pages))
8.                  enqueue(URL_queue, u);
9.             end if
10.           if (crawled_pages.count() % k == 0)
11.                reorder_queue(URL_queue);
12.           end if
13.      end for
14.  end while
```

The enqueue(queue, element) function appends an element to the end of the queue, dequeue(queue) removes the element at the beginning of the queue and returns it, and reorder_queue(queue) reorders the queue using one of the ranking algorithms below:

- Breadth-first

  do nothing
- Backlink count

  for each u in URL_queue
  backlink_count[u] = number of u input links
  sort URL_queue by backlink_count[u]
- PageRank

  solve the following set of equations:

$$PR[u] = \frac{1-0.85}{n} + 0.85 * \sum_{P_i \in B[u]} \frac{PR[P_i]}{O[P_i]}$$

where B[u] shows list of pages linking to u and O[P_i] is the number of links in page P_i sort URL_queue by PR(u)

The OPIC algorithm runs with a different procedure [16]. To be fair, we set OPIC to crawl each page only once but the received cash is changed every time we visit its link in other pages.

The aim of the crawling is to find hot pages. To achieve it, we choose the PageRank algorithm as benchmark. First, the ranks of the entire Web pages are computed using PageRank algorithm on the entire Web graph. At this point a perfect algorithm would have visited pages $r_1 \ldots r_q$, where r_1 is the page with the highest Pagerank value, r_2 is the next highest, and so on. In a set of K pages, gathered from a running algorithm, a page is hot if it exists in the first K hot pages of the benchmark ranking.

Clearly, the algorithm which retrieves the most hot pages will be better than others. We define throughput at each step as a fraction of crawled hot pages to all discoverable hot pages. The performance of the ideal algorithm is of course 1.

We compared the aforementioned algorithms compared to IECA in figures 4-6 on the three different Web graphs. The damping factor of PageRank, β, and δ_t were set to 0.85, 0.1, and 0.35, respectively. As the following figures show, IECA outperforms all other algorithms in the three tested Web graphs. For example, in Fig. 5, when 45% of pages are crawled, IECA finds about 57% of hot pages whereas Partial PageRank, Batch PageRank, FICA, and OPIC find 53%, 51%, 49%, and 51% of hot pages, respectively. In other words, during the crawling process in the UK Web graph, the popular Batch PageRank algorithm and the simple crawling algorithms such as the Backlink count and the Breadth-first algorithms have similar throughputs. Table II shows the improvement of IECA compared to others when 45% of the Web graphs are crawled.

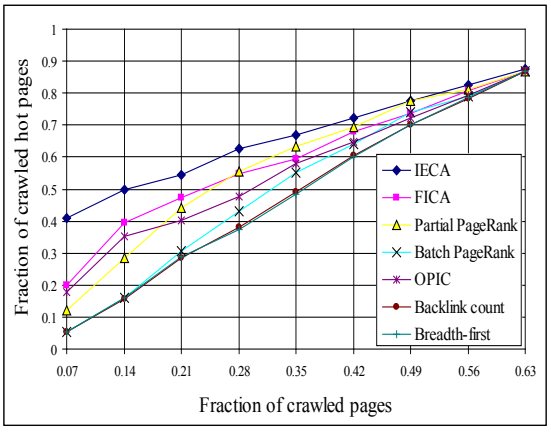

Figure 4. Berkeley -2008 (4 million Web pages).

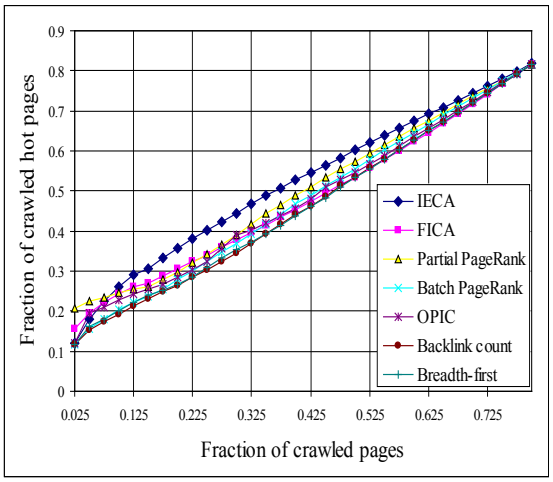

Figure 5. The UK-2005 (10 million Web pages).

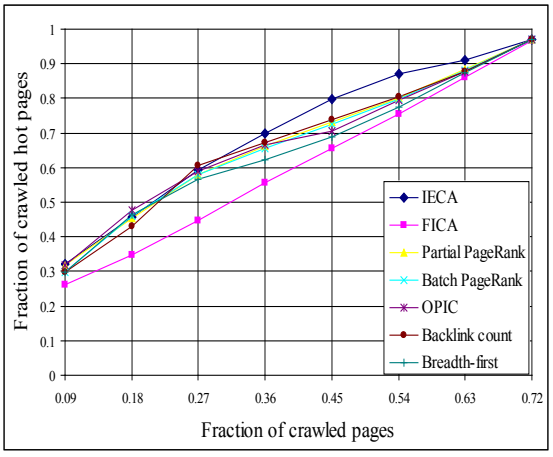

Figure 6. Iran-2010 (3 million Web pages).

V. Complexity of IECA

The complexity of IECA algorithm in the average case is O(E*log(V)).

Proof. The average time of insertion or deletion for a binary tree of size M is log(M). Each edge of the graph is traversed at most once and for each such edge the priority queue will be updated. Thus in the average case, we have E insertion or deletion so that the time of all insertions or deletions is O(E*log(V)). We use a binary tree for queue management, so the complexity of our algorithm in the average case is reduced to O(E*log(V)). While the complexity of PageRank in the worst case is O(V*E) in which V and E are the number of nodes and edges, respectively. However, based on [13] it is around O(100*E) (i.e. 100 iterations for an acceptable ranking is sufficient). In PageRanked-based crawling algorithms (Partial PageRank and Batch PageRank) we run PageRank *K* times and in each step *V/K* nodes will be added to the number of available nodes (discovered nodes). Therefore, in the first run of PageRank we have *V/K* nodes, in the second run we have *2V/K* nodes, in the third run we have *3V/K* nodes and finally in the last run we have *KV/K =V* nodes. Let's suppose by adding *V/K* nodes to the available nodes, *E/K* edges will be added to the available edges. With this assumption the complexity of crawling algorithm based on PageRank is of

$$O(100*E/K + 100*2E/K + 100*3E/K + \ldots + 100*E) = O(50*E*(K+1)).$$

Thus, IECA is faster than crawling algorithms based on PageRank by a factor of (50*(K+1))/log(V). For example, if we have 10 billion pages ($V=10^{10}$) and we run the PageRank algorithm for 200 times (*K=200*), the complexity of any crawling method based on PageRank is O(10050*E) whereas IECA complexity is O(10*E). Moreover, IECA has low memory complexity and it only needs a distance vector of Web pages O(|V|). We also experimentally verified the IECA complexity. Fig. 7 shows a Web graph from the crawler view. The Web graph from the crawler view is divided into three parts as the following:

- **First part**. Crawled Web pages. It includes the pages that are downloaded by the crawler.

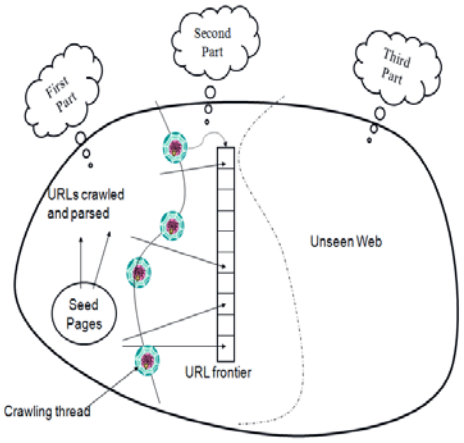

Figure 7. The Web graph from the Crawler view.

- **Second part**. Discovered Web pages. List of URLs that are visited by the crawler and are inserted into the URL frontier, a priority queue containing pages waiting to be crawled.
- **Third part**. Undiscovered Web pages, the pages that their URLs are not visited by the crawler.

In IECA the Web graph is traversed only once and only second part of the Web graph is participated in the importance computation of the Web pages, unlike the PageRank based algorithms, they use the first and second parts of the Web graph in the importance computation in each iteration. To clarify the issue, IECA and PageRank based algorithms are compared in terms of the number of the nodes in the memory in different snapshots of the crawling process. Fig. 8 shows the obtained results of the experiment.

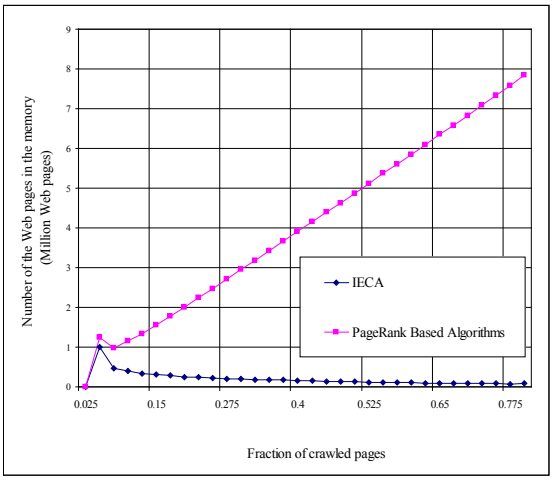

Figure 8. Number of the Web pages in the memory in different snapshots , UK Web graph (10 million Web pages)

According to Fig.8, when 52% of the Web pages are crawled, IECA, Batch PageRank, and Partial PageRank would need 200,000, 5 millions, and 5 millions Web pages for page importance computations, respectively.

VI. MAJOR CONTRIBUTIONS OF THE PAPER

The main contributions of the paper are:

1. This paper introduces an algorithm called IECA (Intelligent Effective Crawling Algorithm), whose goals is to find more important pages faster in the crawl in comparison with crawling algorithms presented in the literature.
2. In IECA links are selected according to metrics such as link weight (based on out-degrees), logarithm distance, and P-C coefficient, but introduces the use of reinforcement learning to balance these factors during the progress of the crawling in an adaptive way.
3. We tested IECA using several real Web graphs and show that it achieves the objective of downloading important pages early in the crawl. We also evaluated of IECA on the Web graph of Iran.
4. IECA is easy to implement with low time O(E*logV) and memory complexity O(V)-V and E are the number of nodes and edges in the Web graph, respectively.
5. IECA with less complexity has higher performance compared to previous algorithms.
6. IECA does not need to save the matrix of Web graph, and only a vector of Web graph nodes for saving the distance of pages is enough.
7. IECA traverses the Web graph only once and it does not have any iteration during the crawl.

VII. CONCLUSION AND FUTURE WORK

In this paper we proposed a new crawling algorithm called IECA. This algorithm selects each page based on its background knowledge from visited pages and the feature of the Web environment. In fact it makes a relationship between the obtained knowledge by the crawler and the Web structure feature. For evaluation of IECA we used three different Web graphs (the UK-2005, Berkeley-2008, and Iran-2010). The results show IECA is an efficient crawling algorithm with low time and memory complexity.

There are two directions in which we would like to extend this work. One direction is to execute IECA as a crawling algorithm on a dynamic Web graph and the second direction is to evaluate IECA as a ranking algorithm.

ACKNOWLEDGEMENTS

This research was supported by ICT Research Institute (ITRC).

The authors also would like to thank Seyyed Ali Hashemi for valuable comments.

REFERENCES

[1] A. Gulli and A. Signorin, "The indexable web is more than 11.5 billion pages," *in 14th international conference on World Wide Web*, 2005.

[2] C. Olston and M. Najork, "Web Crawling," *Foundations and Trends in Information Retrieval*, vol. 4, no. 3, pp. 175-246, 2010.

[3] S. Batsakis, E. G.M.Petrakis, and E. Milios, "Improving the performance of focused Web crawlers," *Data & Knowledge Engineering*, vol. 68, no. 10, 2009.

[4] M. Najork, J. Wiener, "Breadth-First Search Crawling Yields High-Quality Pages," *in 10th International conference World Wide Web,* 2001.

[5] H. Ali, "Effective Web crawlers," *School of Computer Science and Information Technology,* Ph.D thesis, Melbourne, Victoria: RMIT University, 2008.

[6] J. Cho, "Crawling The Web: Discovery and Maintenance of Large-Scale Web data," *Computer science. Stanford University*, Ph.D thesis, 2001.

[7] A. ZarehBidoki, N. Yazdani, and P. Ghodsnia, "FICA: A novel intelligent crawling algorithm based on reinforcement learning," *Web Intelligence and Agent System*, vol. 7, no. 4, December 2009.

[8] J. Kleinberg, "Authoritative Sources in a Hyperlinked Environment", *Journal of the ACM (JACM)*. Vol. 46, no. 5, September 1999.

[9] L. Page and S. Brin, "The PageRank citation ranking: Bring order to the Web," *Technical report, Stanford University*, 1998.

[10] S. Brin and L. Page, "The anatomy of a Large-Scale Hypertextual Web search engine", *in Seventh International conference on World Wide Web*, 1998.

[11] S. Pandey and C. Olston, "User-centric Web crawling," *in 14th international conference on World Wide Web*, 2005.

[12] J. Cho and U. Schonfeld, "RankMass Crawler: A Crawler with High Personalized PageRank Coverage Guarantee," *in*

33th International Conference on Very Large Databases 2007.

[13] T. Haveliwala, "Topic-Sensitive PageRank," *in 11th International Conference on World Wide Web*, 2002.

[14] Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen, "Combating Web spam with TrustRank," *in 13th international -conference on Very large databases*, 2004.

[15] J. Cho, H. Garcia-molina, and L. Page, "Efficient Crawling Through URL Ordering," *in 7th conference on Word Wide Web*, 1998.

[16] S. Abiteboul, M. Preda, and G. Cobena, "Adaptive On-Line Page Importance Computation," *in 12th international conference on World Wide Web,* 2003.

[17] C.Castillo, M. Marin, A. Rodríguez, and R. Baeza-Yates, "Scheduling algorithms for Web Crawling," *in Latin American Web Conference WebMedia*, 2004.

[18] A. Dasgupta, A. Ghosh, R. Kumar, R., C. Olston, S. Pandey, and A. Tomkins, "The discoverability of the Web," *in 16th international conference in World Wide Web*, 2007.

[19] A. ZarehBidoki and N. Yazdani, "DistanceRank: An intelligent ranking algorithm for Web pages," *Information Processing and Management,*, vol. 44, no. 2, 2008.

[20] M. Dikaiakos, A. Stassopoulou, and L. Papageorgiou, "An investigation of Web crawler behavior: characterization and metrics," *Computer Communications*, vol. 28, no. 8, pp.880-897, 2005.

[21] R. Baeza-yates and C. Castillo, "Crawling the Infinite Web," *Journal of Web Engineering*,vol. 6, no. 1, 2007.

[22] Y. Matsuo, Y. Ohsawa, and M. Ishizuka, "Average-clicks: A New Measure of Distance on the World Wide Web, " *Journal of Intelligent Information Systems*, 2003.

[23] R. Sutton, and A. Barto, "Reinforcement Learning: An Introduction," MIt Press, Cambridge, 1998.

[24] P. Boldi, and S. Vigna, "The Web graph framework I: compression techniques," *in 13th international conference on World Wide Web*, 2004.

[25] Web Graphs, http://Webgraph.dsi.unimi.it/

[26] P. Boldi, M. Santini, S. Vigna, "Do Your Worst to Make the Best: Paradoxical Effects in PageRank Incremental Computations", *In Proceedings of the third Workshop on Web Graphs (WAW),* 2004.

[27] R. Baeza-yates , C. Castillo , M. Marin , A. Rodriguez , "Crawling a Country: Better Strategies than Breadth-First for Web Page Ordering" *In Proceedings of the 14th international conference on World Wide Web,* 2005.

Mohammad Amin Golshani received his B.Sc. degree in computer engineering from Department of Electrical and Computer Engineering of Zanjan University, Iran, in 2009 and his M.Sc. degree in Information Technology (IT), under the supervision of assistant professor ZarehBidoki in Yazd University, Yazd, Iran, 2011. His current research interests include Web crawling, Web ranking, and spam detection.

AliMohammad ZarehBidoki got his B.Sc. degree in computer engineering Isfahan University of technology in 1999. He also received M.Sc. degree in Computer Architecture from University of Tehran in 2002. Furthermore he received his Ph.D. degree in Computer Engineering at University of Tehran in 2009. His research interests include Web Information Retrieval, Search Engines and Data Mining. He has published more than 20 papers in international journals and conference. Currently, he is an assistant professor in Yazd University.