Focused Web Crawling using an Auction-based Economy

ERIC B. BAUM†, ERIK KRUUS, IGOR DURDANOVIC and JOHN HAINSWORTH*
NEC Research Institute

Our aim is to produce a focused crawler that, given one or a number of sample pages, will crawl to all similar pages on the web as efficiently as possible. A key problem in achieving this goal is assigning credit to documents along a crawl path, so that the system can learn which documents lead toward goal documents and can thus efficiently search in a best first manner. To address this problem, we construct an artificial economy of autonomous agents. Each agent buys and sells web pages and is compensated when it buys a goal page, when another agent buys the current set of uncrawled web pages, or when future agents buy a goal page. The economy serves to push money up from goal pages, compensating agents that buy useful pages. Inappropriate agents go broke and new agents are created, and the system evolves agents whose bids accurately estimate the utility of adding pages to the search. The system is found to outperform a Bayesian focused crawler in our experiments.

Categories and Subject Descriptors: I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent Systems*; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Search Process*; I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods and Search—*Graph and tree search strategies*

General Terms: Algorithms, Economics

Additional Key Words and Phrases: agents, auction, economy, focused web crawling, Nash equilibrium, www

1. INTRODUCTION

With established search engines indexing only about 15% of the web [Lawrence and Giles 1999], many authors have realized the need for topic-driven crawlers that can be taught to find relevant pages with modest resources, c.f. [Steele 2001; Ester et al. 2001; Chakrabarti et al. 1999]. A *focused crawler* takes as input one or several related web pages and attempts to find similar pages on the web, typically by recursively following links in a best first manner. Ideally, the focused crawler should retrieve all similar pages while retrieving the fewest possible number of irrelevant documents.

At any given point in a best-first crawl, there is a set of pages already fetched, which we call traversed pages, and a set of linked pages from which one must select the next page to fetch. We call this frontier the *corona*. A variety of mechanisms have been proposed to rank the pages in the corona, thus defining "best" in one's best-first crawl, including page similarity and popularity [Pant et al. 2002], evolutionary sets of neural networks [Pant and Menczer 2002], Bayesian probabilities [McCallum and Nigam 1998],

Address: NEC Research Institute, 4 Independence Way, Princeton NJ 08540 (USA);
(†) Corresponding author
(*) Princeton University, 35 Olden St., Princeton NJ 08540, hains@cs.princeton.edu

other word-counting heuristics (document count, term frequency (TF), term-frequency-inverse-document-frequency (TF-IDF) [Lewis 1992]), maximum entropy [Nigam et al. 2000], reinforcement learning [McCallum et al. 1999; Menczer and Monge 1999], information gain [Glover et al. 2002], goal-connectivity (context graphs) and back-crawling [Diligenti et al. 2000], and semantic content [Diao et al. 2000].

As was noted by Rennie and McCallum [1999], the key problem is one of assignment of credit. One must decide which links will lead to the best overall crawl. Optimization may require sacrificing short term gains in order to follow links that lead in the direction of multiple goal pages. One must be able to identify ancestors (grand parents, great grand parents, etc.) of goal pages even though these may be on rather different topics than the goal pages themselves. For example, to find papers on neural networks, one might wish to find home pages of computer scientists, and to find these one might wish to reach home pages of computer science departments [Diligenti et al. 2000].

In this paper we apply the Hayek Machine learning framework [Baum 1996; Baum and Durdanovic 2000b; 2000a] to this problem. Hayek simulates an artificial economy of agents that bid to take actions. The economy is constructed so as to assign credit to the individual agents for their contribution to the performance of the system. As some agents go broke and are removed from the system, and as new agents are created, the system evolves, learning accurately to assign credit to intermediate states, and learning to chain long sequences of agents to solve hard problems. In previous work, it has solved problems in Blocks World, Rubik's Cube, and Rush Hour domains involving credit assignment through sequences of many hundreds of actions. Because the economy is a framework for handling assignment of credit among agents, the Hayek Machine framework is flexible enough to incorporate agents using many different ranking mechanisms that cooperate on a single focused crawl.

Section 2 describes the Hayek Machine and its application to focused crawling. Section 3 gives empirical results for Hayek and compares these to a simple Bayesian best first crawler modeled on [McCallum and Nigam 1998] and to breadth first search. Section 4 describes some possible extensions of the model. Section 5 concludes.

2. THE ARTIFICIAL ECONOMY

Our economy consists of a learning machine, which we call the Hayek Machine, that interacts with a world that it may sense and take actions on, and which makes payoffs when put in an appropriate state. For the web crawling domain discussed here, the world consists of the web, and we make payoffs to the system whenever it retrieves a goal page. The idea is that if the economy is set up correctly, money will flow into the system when it achieves goals, and will flow through the system to agents that helped the system reach these goals.

In this paper, we assume that goal pages can be recognized when retrieved and thus focus on the problem of learning an efficient focused crawl. This follows the previous work of Diligenti et al. [2000], who implicitly assumed that goal pages were sufficiently similar to the user-submitted example pages that they would contain a sufficient number of key words to be readily identifiable. We discuss the problem of identifying goal pages at greater length in section 3, but for now note simply that the Hayek machine, as formulated here, must be given a reward when it retrieves a goal page by some auxiliary program (or possibly human feedback).

The Hayek machine consists of a collection of agents, each consisting of a computer program with an associated numeric "wealth". The system acts in a series of auctions. In each auction, the agents evaluate their programs, returning a bid. The high bidding agent buys the right to take actions. It pays the auction price to the previous winning agent, takes its action, collects any reward paid by the world, and then sells the right to take action in the next auction. In the present paper, the action taken by the winning agent is simply to add a page to the traversed set.

New agents are periodically created, typically as mutations of existing wealthy agents, and added to the system. Also agents whose money falls below their initial allotment are removed. A small tax is occasionally assessed, to remove inactive agents. The population of agents, and thus the performance of the system, evolves.

The auction protocol allows discovery of long chains of agents that accurately value states. In any auction, the high bidding agent will tend to be one that takes the world toward a state of reward, and whose bid accurately estimates its expected pay-in. An agent that does not take the world toward payoff will not be profitably able to bid high, and an agent whose bid does not accurately estimate its expected pay-in will either die or soon be outbid by a mutated version of itself that bids higher.

Such lengthy chaining has been observed in previous experiments on Blocks World, Rubik's Cube, and the commercially sold game of Rush Hour, where collections of agents were evolved that accurately valued states for hundreds of actions or more before achieving reward, c.f. [Baum and Durdanovic 2000b; 2000a]. If conservation of money and a certain notion of property rights are enforced, then money flows in from achieving goals, and is pushed back to compensate enabling agents. In those references and [Baum 1998] we discuss at greater length the economic principles that stabilize performance of the system.

Note that, at least in principle, such auction protocols are capable of compensating agents for multiple contributions. So, for example, an agent that adds a web page to the search which has links pointing to several later goal pages could be compensated for the total expected pay-in to the system. Note also that the protocol makes no insistence on how the agents decide what to bid or what actions to take. Thus the evolution of the system could in principle forge cooperation between agents using widely different criteria for deciding when to bid or what action to take. For example, agents could use widely different ranking mechanisms for deciding whether to add a page to the search set.

2.1 Hayek and Web Crawling

This section will describe the specific protocol we used to train Hayek on focused crawling in the experiments reported. Non-critical features have been simplified.

Algorithm 1, Train_Hayek_Crawler, outlines the main steps used to train the set of agents in the most common run configurations. A series of auctions and the actions taken by auction winners create a focused web crawl, as detailed below. The system is trained using a number of short auction sequences. Note that the system has no guidance if it is not earning reward. Hence it is necessary to begin training with relatively simple examples. Thus we begin by doing a back-crawl[1] from goal pages, and start the training at pages known to be one link from goal pages.

We support crawling where all links of a page are automatically fetched, as in [Diligenti

[1] A back-crawl from a web page is done by submitting the URL of the page as the search string to a search engine. Ad pages are removed from the result by checking the contents of each page found to make sure it points to the specified URL.

Algorithm 1 Train_Hayek_Crawler

```
Do a limited Google backcrawl to set up initial goal sets and connectivity information
[opt.] read_goalconnected_URLs()
obtain_initial_set_of_agents()
for auction_sequence=0..N do
  difficulty = Select_Problem_Difficulty(difficulty, MA, hystlo, hysthi,):
  (traversed_set, corona, number_of_auctions) = Select_Problem(difficulty, connectivity)
  repeat
    (best_agent, url) = Choose_Highest_Bidder(corona, agents)
    if best_agent chose a virtual url from the corona /* resell url immediately */ then
      virtual_agent = best_agent
      Fetch_Web_Content(url)
      best_agent = Choose_Highest_Real_Bidder(url,agents)
      if best_agent == root_agent, then best_agent = New_Real_Agent(url)
      [opt.] Pay_Immediately(best_agent,virtual_agent,...)
    end if
    if best_agent == root_agent, then best_agent = new_real_agent(url)
    if best_agent is a real agent /* transform the world */ then
      insert into corona all links from url not matched by entries in traversed_set
      [opt.] fetch content of these links
    end if
    add url [opt. entire site of url] to traversed_set,
    mark url in traversed_set as owned by best_agent
    delete entries matching traversed_set from corona
    [opt.] Pay_Immediately(best_agent,...)
  until corona is empty or number_of_auctions have run or [opt.] url is a goal page
  if payments were not made immediately, then pay_afterward(auction_history)
  collect tax, remove childless unprofitable agents, allow wealthy agents to spawn
  [opt.] Forget_All_Real_Web_Content() /* train virtual agents too */
end for
save_Hayek_agents();
save_goalconnected_URLs();
```

et al. 2000], as well as crawling where virtual agents, bidding on hyperlink information, select individual pages to retrieve, as in [Ester et al. 2001; Rennie and McCallum 1999]. Thus two distinct types of pages can exist during the crawl. Initially a page has a URL and one or more associated hyperlink contexts. These hyperlinks and their context define a set of virtual features; i.e. features which describe a page but are available before the page has been fetched. A page with only virtual features is a "virtual" document and may be selected by virtual bidders who base their bid on their estimate of the desirability of a page having those features. After content has been fetched one has a "real" document, with real features (e.g. words), as well as virtual features. Real bidders, whose bids depend on the presence of one or more real features, may then be able to submit nonzero bids for these pages.

At any point in the forward crawl, there is a *traversed_set* of pages that have been visited, and a set of pages in the *corona*, namely pages linked by pages in *traversed_set*. Each agent in the population offers a bid to add some page in *corona* to *traversed_set*. This bid will be zero if the agent does not wish to buy the page. The bids are maintained in an ordered queue. In each auction, the highest bid is popped off the queue, the associated *best_agent*

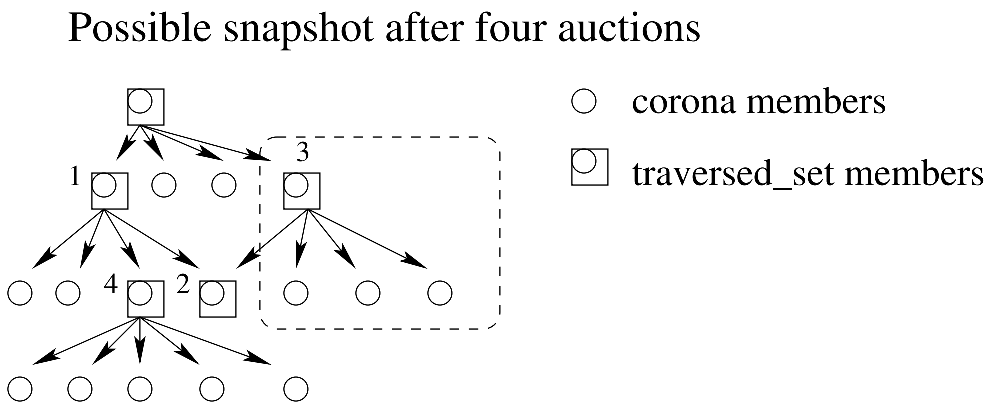


Fig. 1. Possible snapshot of page connectivity after four auctions of a forward crawl. Arrows show all hyperlinks found in the *traversed_set* web pages. Changes arising from the third auction are in a dotted box. This auction has moved one *corona* page into the *traversed_set* and 3 of the 4 hyperlinks in this page have added new destination URLs to the *corona*.

wins the auction, and the associated page P is added to *traversed_set*. Pages linked by P are added to *corona*. The winning *best_agent* pays its bid to the previous winner. If page P is a virtual page, the *best_agent* must fetch web content for the page. If page P has web content and is a goal page, the *best_agent* receives a reward. Lines 9-15 of algorithm 1 force immediate resale of virtual urls after their content has been fetched in order to ensure that virtual agents receive proper feedback about their choices before the training run terminates.

Figure 1 shows a possible snapshot showing the *traversed_set* and *corona* that have been constructed after four auctions. Here the virtual pages are marked only with circles: hyperlink information exists for them, but no page content has been retrieved. Squares mark pages with additional real content.

Algorithm 2 Select_Problem_Difficulty (**out:** *difficulty*; **in:** *MA, hystlo, hysthi*)

/* *MA* indicates cumulative successfulness and increases or decreases by a small step depending on whether the previous auction sequence encountered a goal page or not. */
if *MA* > *hystlo*×*difficulty*, **then** ++ *difficulty*
if *MA* < *hysthi*×*difficulty*, **then** −− *difficulty*

If the system manages to consistently find goal pages during auction sequences, the problem difficulty is increased (algorithm 2, Select_Problem_Difficulty) and training begins with pages whose estimated distance to a goal page is higher. In practice, it was found beneficial to dither between two distances during the transition to higher difficulties, as governed by parameters $hystlo$ and $hysthi$.

To select a problem, a random site was first chosen, and then a URL from that site, as described in Algorithm 3, Select_Problem. Back-crawling was only used to set up the initial goal-connected set. Thus the goal-connected set, and the pool of initial problems, grew slowly in the crawls reported here. To guard against continually retracing the same similar set of paths to reward, a lengthier auction sequence was forced every 500 auction sequences.

Algorithm 3 Select_Problem

input: *difficulty*, auction sequence number, goal *connectivity* information
output: *traversed_set, corona, number_of_auctions*

clear *traversed_set*, erasing old ownership information.
repeat
  candidates = set of URLS whose best known connectivity to a goal page is *difficulty*
  if too few *candidates* **or** too many attempts, **then**
    candidates = randomly choose **either** URLs of unknown connectivity, **or** the set of one-away URLs
  end if
  select a random site from those in *candidates*, then random *url* from that site.
until *traversed_set* defined by *url* has enough links to form a valid initial *corona*.
[opt.] fetch web content for the initial *corona*
Limit auction sequence to $number\_of\_auctions = 2 \times difficulty + offset$.
/* Intermittently substitute *offset* [defaults to zero] with a large value and set a flag to not stop at the first goal page during the following auction sequence. */

Algorithm 4 Obtain_Initial_Set_of_Agents

/* Generate a unique *root_agent*, that bids zero for everything and takes default action in case no other agent bids on a corona or URL, and that supplies a default mechanism to spawn new agents. */
if restarting **then**
  read_Hayek_agents() /* with complete bid information */
else
  By default, select words from a page or dictionary according to information gain.
  Other weighting schemes available were uniform, TF, TF-IDF, document count, information gain, unbalanced correlation score (CORR_ub) [Weston et al. 2002] and differential CORR_ub.
  Initial bids of these agents are indeterminate, marking them as *explorers*.
end if
Attempt to assign a URL containing the words used for an agent as the "parent" url for the *agent*. /* This URL will be used for mutating words of sufficiently wealthy agents */
[opt.] Generate some virtual agents, using connectivity, URL or link context information in conjunction with word weights

Our bidding agents were defined by a discrete set of features, all of which had to be present for the agent to submit a bid. Algorithm 4 (Obtain_Initial_Set_of_Agents) or a variant is used to create the initial agents. The features defining the agent were chosen randomly from a given page using a probabilistic weighting based on one of a number of algorithms. Weightings we programmed included uniform, term frequency (TF, i.e. # of times a word appears in some set of URLs), term frequency - inverse document frequency (TF–IDF), document count (CORR_ub, # of documents containing the word) [Lewis 1992], differential document count[2] and information gain [Glover et al. 2002]. Typically, one

[2]Document count was normalized to the range [0,1], following the unbalanced correlation method (CORR_ub) used with considerable success in sparse recognition problems by [Weston et al. 2002]. The differential document count looked at the difference of this normalized word occurrence between sets of connectivities n and $n-1$ to derive weights for words. It was reasoned that such agents should be particularly good in context n at driving the system closer to goal pages.

such weighting algorithm was chosen at random and the agent was then created using that weighting to select the features. For example, when TF was used, words were chosen from the page to be used as features with probabilitiy proportional to TF. In some runs, we used only a single weighting scheme, as will be noted when we discuss empirical results.

In addition to the page-recognizing agents, a "root agent" was always available, and bid zero in every auction. The root agent won any auction that did not attract a positive bid, and when it won took the action of creating a new agent using features from some page in the *corona*, so that this new agent was guaranteed to match this page. The root then resold the world to this newly created agent. The system assigns ownership to URLs in the traversed set (algorithm 1, line 22) which are used for compensation as discussed below (c.f. algorithm 6, line 4.) The root agent owned the URLs of pages in the initial *traversed_set*.

Each agent of wealth greater than an arbitrary threshold of ten times the goal page reward used a procedure like algorithm 4 to create a new agent that is like itself but with features added, deleted, or changed. Each non-root agent was assigned a parent agent which gave it initial capital (so that it could bid) in exchange for a fraction of its profits, (as discussed below, c.f. algorithm 6, line 5). In addition to single words, agents could choose features based on words in the URL string (before, at, or after the site words), words near the hyperlink text (3 words before, in, or 3 words following), and estimated goal connectivity. When all features matched, an agent would bid.

At the core of the web crawl lies the priority-queue auction bidding in Choose_Highest_Bidder (algorithm 5). We experimented with two different methods (algorithm 5, lines 17-21) of assigning bids to agents. One method, described further in section 3.2, sets each agent's bid to a fixed value at the first auction in which it can bid. This is called the bid-epsilon model because this bid is set to a value slightly higher than any competing bid, i.e. the new agent is forced to submit a winning bid. As new agents were created, bids were thus pushed up toward the highest profitable level.

The other method, called the bid-estimating method, is described in more detail in section 3.3. Here we used a second-price auction, and agents estimated a Nash equilibrium bid. Bid-estimating agents continually maintained and revised their bids.

A clear distinction is maintained between virtual bidders, pages and features versus their real counterparts. When virtual documents are used, adding new links to the crawl frontier has two steps: (i) a virtual bidder buys some hyperlink, based on virtual features, and transforms the world by fetching web content for this page; (ii) a real bidder buys the page now that real features are available.

The real bidder presumably has a better idea of the true worth of the page, and is able to collect any goal page reward. The virtual bidder receives its positive feedback for a good page by a high resale price, as well as any fractional reward money that is allowed to pass back up the auction chain. To force equitable training of the virtual agents in short auction sequences, a virtual agent's page was always resold immediately, and if no real agent submitted a bid, a new agent was created that matched and immediately bought the new page at a low price.

Real web content could be fetched at several points in the algorithm. We chose to use a stopword list (removing short words like *the*) but did not include stemming (so *beach* and *beaches* were separate features). Some important side effects of fetching real web content were:

Algorithm 5 Choose_Highest_Bidder (*corona*, *agents*)

```
output: single best_agent, best_url, seller_agent
/* the full priority queue was maintained only when necessary */
for all agent in agents do
    for all url in corona do
        if all features defining agent are in url, then insert (bid, agent) into priority queue
    end for
end for
obtain the set of explorers (agents of indeterminate bid) and set of highbidders
for all agent in explorers ⋃ highbidders do
    determine all urls in corona matched by agent
end for
/* generate a payoff[] matrix for matched urls */
for all urls in corona matched by any agent in explorers ⋃ highbidders do
    Exploratory_Auction(explorer, corona, url) /* only real bidders bid here */
    estimate payoff[url] using goal page reward, sell price and [opt.] worth of links in url)
    undo any corona changes
end for
for all explorer in explorers do
    if bid-ε agent set, then
        explorer.bid = highbidders.bid + ε
    else
        use payoff[] matrix to set all initial explorer bids
    end if
    allow explorer to supplant or augment highbidders it they bid high enough
end for
best_agent = random selection amongst highbidders
best_url = select an URL matched by best_agent
seller_agent = random selection amongst owners of pages linking to best_url
Allow other highbidders to use the payoff matrix to estimate their own potential earnings and
thereby refine their own bids. No money changes hands during such virtual payoffs.
```

—all web content and virtual information was counted in a detailed set of dictionaries, and dynamically updated at every change;

—virtual link information and estimated connectivities of any URL in the system could also change whenever web content was fetched;

—in principle subsequent bids of any agent, for any page, could change.

Payment could be done immediately (after every auction) or after an auction sequence. The sequence of payment followed the steps outlined in algorithm 6, Pay_Immediately. The agent winning the auction paid the previous agent, and fetched a page. If the agent matched multiple-pages, one was chosen randomly to be added to the traversed set.

The winner of an auction has three other agents that might be deserving of some additional reward: the parent that spawned the winner, the agent that previously owned the world, and a seller agent that owned a page with a hyperlink to the page the winner selected. Three fractions (P, Q, and S respectively) governed such reward-sharing. In the economic model, given enough auctions, resale prices alone will eventually be enough to train a Hayek system. We will note when P, Q or S were non-zero in the empirical runs.

Payments were used to establish a *valuation* for the bought page P. A bid-estimating

Algorithm 6 Pay_Immediately (. . .)

input: *agent, seller, agents, auction_history, reward*. . . three [0,1] kickback fractions: *P, Q, S*
output: *agents* {all of which could potentially have gained or lost money}
/* Throughout transaction steps, separately track *profit* (based on money that actually changed hands) and *valuation* (based on money that should have been paid). First- and second-price auctions treat *valuation* differently. The final *valuation* is filtered to produce an agent's *reward_factor*. A self-estimating agent's *reward_factor* is used to set a reasonable bid. */

pay auction price to previous auction winner in *auction_history*
collect any goal page *reward*
pay $Q \times reward$ to the list of previous auction winners
/* payments form truncated geometric series */
pay $S \times reward$ to the *seller* agent
evaluate net *profit* pay $P \times profit$ to parentage of *agent*
/* payments form truncated geometric series */
if using self-estimating agents, **then**
  agent updates its time-discounted *reward_factor* to approach this *valuation*
  agent bid is the *reward_factor*, less up to 2% for agents that are older /* Ties favor new agents */
  /* *P, Q* and *R* kickbacks also tweak the *reward_factor* maintained by those being paid */
end if

auction winner updated a slowly-varying estimate of the *valuation* of P. The estimated *valuation* was used to set the agent's subsequent bid amount. For training purposes, more experienced agents were assigned a small (up to 2%) bid reduction. This allowed "equally good" new agents to refine their bids into realistic estimates of worth more quickly.

3. EMPIRICAL RESULTS

We experimented on the following two domains: finding beach volleyball pages and finding corporate officers. Goal pages in the beach volleyball domain were defined as pages including the words {beach, volleyball, fivb, behar}. A points-based judge of several commonly co-occurring words was used to define the board-of-directors goal page judge. The volleyball problem was one which involves around 180 goal pages in a reasonably well connected set of urls. A difficulty with this problem was that huge number of sports-related pages could easily trap the system. Hockey, football, baseball and golf pages were particularly prevalent within the web pages fetched.

We begin with three subsections that compare algorithms that fetch all links from any chosen page automatically. This approach avoids the complexity of virtual documents, and of how bids based on hyperlink content and surrounding text may affect different learning algorithms. Section 3.1, presents results from standard Bayesian best-first and breadth-first search algorithms that we use as the standard by which further work is judged. Figure 6 below directly compares several variants of the Hayek scheme to Bayesian focused crawlers.

Later we will compare these runs with systems that do not automatically fetch all links, but develop "virtual agents" that bid upon virtual documents to select unfetched but linked-to pages based upon virtual information only. All runs in this paper started off with very few goal pages (15-18 for volleyball and 15-45 for board-of-directors) as returned from a Google search and back-crawl. The back-crawl results determined the initial word and document statistics, as well as supplying the initial selection of problem URLs.

In all Hayek runs in this paper, the number of agents in the population gradually increased. Typically the run had fetched most of the goal pages by the time the population consisted of a few hundred agents, but then, with multiple profitable agents spawning offspring, the population continued to increase. We capped the total number of agents at between one and five thousand, depending on the run, and after the cap was reached refused to allow creation except when other agents went broke and were removed.

3.1 Bayesian and Breadth-first reference crawls

We first describe results with standard Bayesian and Breadth-first methods that we will then use as benchmarks to which we compare the Hayek results. Bayesian and breadth-first searches are often used as reference points to compare the efficiencies of focused crawlers, c.f. [Rennie and McCallum 1999; Menczer and Monge 1999; Diligenti et al. 2000].

The naive Bayesian crawl uses the probability that a page with a given set of features is in the set of goal pages to decide which page to follow in a best-first manner. For this purpose, one needs statistics giving the frequency with which words appear in goal pages, and the frequency with which words appear in general pages. We thus kept continually updated individual word and document counts. We followed the formulation of McCallum and Nigam [1998] in calculating these probabilities using either binary word presence or the actual word counts in pages to derive the estimate. Our Bayesian crawler was thus similar to the standard focused crawler used as a comparison in [Diligenti et al. 2000] except that our Bayesian crawler was potentially more powerful in that it updated its statistics during the crawl as more goal pages were discovered. As is typical in web crawls, figure 2 shows that a multinomial model which took into account the number of times a word appeared in a page fared better than the binomial (present-or-not) model. All links of selected pages were fetched and no virtual document features were in the dictionaries for Bayesian runs.

Often fewer than 50 goal pages were found within the first 20000 pages fetched. Continuing these runs typically required well over 100000 fetches before improving and finding over 100 goal pages. Breadth-first searches required a similarly large number of web fetches before retrieving a large number of goal pages. A breadth-first search starting at a random one-away URL fared on par with the Bayesian methods for this well-connected problem, and eventually picked up 180 goal pages after about 150000 page fetches. Ordinarily one expects breadth-first searches will fare much worse than Bayesian methods (e.g [Diligenti et al. 2000]), but for the beach volleyball problem, generic sports pages were statistically similar enough to fool Bayesian agents trained on a relatively small number of goal pages.

3.2 Bid-ε Agents

We next describe results using the Hayek system with bid-ε agents. The bid-ε model determines the bids of agents by forcing the first-time bids of new agents to be incrementally higher than the highest competing bid. Thereafter, an agent's bid remains fixed at this initial value. An agent whose initial bid was too high was immediately unprofitable and thus removed. As mutated and cloned versions of profitable agents are created, bids are pushed

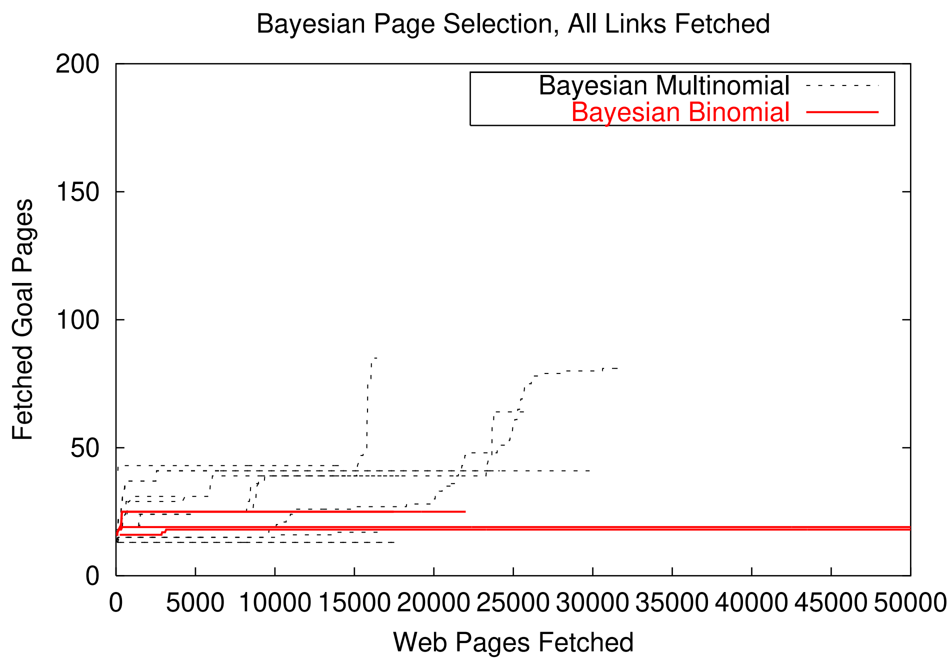


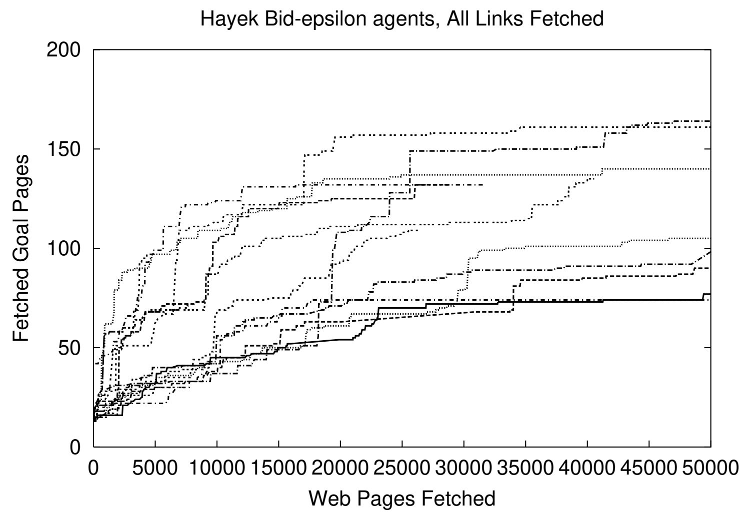
Fig. 2. Tracks show individual runs where Bayesian probability of being a goal page was used to choose links to follow. All links from a chosen page were fetched. No virtual documents were used.

upward toward accurate estimates of value.

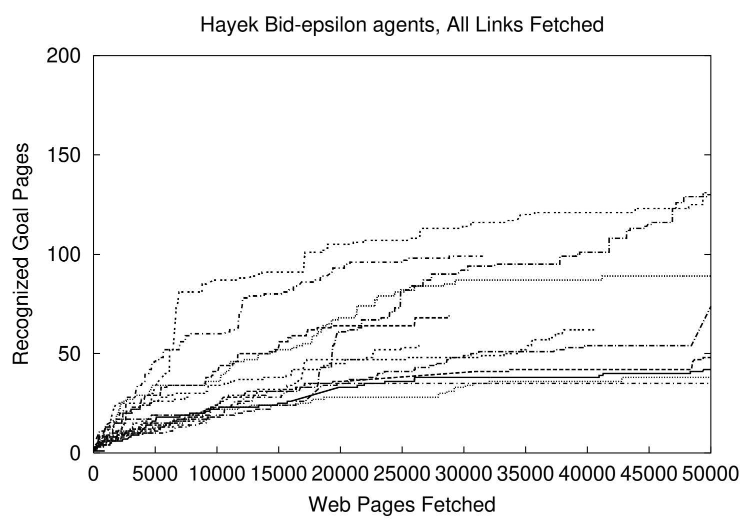
To create agents, we added a procedure that used a statistical score to select words from seed urls to create or mutate agents (see description of algorithm 4). The system typically used arbitrarily-assigned probabilities to select a weighting method, and then create agents. Bid-ε agents used uniform, TF, TF–IDF, document count or differential document count rankings. The differential document count and TF-IDF rankings appeared to generate the most reasonable agents for the volleyball problem.

Figure 3 shows the behavior of such agent sets with no virtual documents. While the bid-ε agent sets generally continued to improve steadily over the first 50000 web fetches, there is large variation in the learning rate. By fetching all links of every selected url, we can directly compare the performance of these agents with the results of the naive Bayesian crawler. The appropriate comparison is between figure 3(a) and figure 2, both of which report the number of goal pages fetched. We observe that becoming "trapped" in large non-goal communities was less of a problem than with the Bayesian agents of figure 2. It seems that the potential to radically change one's bidding behavior is particularly helpful in learning how to deal with training data that supplies few positive examples.

The reason for the variation in learning rate was that good agents disappeared, often to be replaced by worse, but still profitable agents, before a similar agent gets created later on to correct the situation. Eventually, the bid-ε set of agents would correct a bad set of agents and improve it, but this was a slow process. We also observed that the bids of the best, usually-successful bidders in the runs of figure 3 were still much below the constant goal page reward that we used. This indicates that the bid-ε agent sets were



(a) Fetched goal pages use a perfect goal page recognizer as soon as any pages are fetched and added to the corona.



(b) Recognized pages are those actually selected by some agent during an auction

Fig. 3. Each track shows an individual training run, plotting fetched (a) and recognized (b) goal pages. The Hayek Bid-epsilon method was used to adapt agent bid values.

still far from steady-state bids, even after several thousand short auction sequences. The mutation-directed, bid-ε approach of an agent's bid to a local equilibrium value was too slow to be practical.

Comparing the two graphs, 3 (a) and (b), one sees that there was often a lag between when goal pages were fetched and added to to the corona and when these same pages were actually recognized (i.e. added to the *traversed_set* by a winning agent.) This occurs because these auction sequences were short and preemptively terminated at the first selected goal page. Slower training (alg. 2) and occasional longer auction sequences (alg. 3, line 11) could minimize the discrepancy between the number of fetched and recognized goal pages.

The main effect of a slower training was to have the system retrace tiny variations of the auction path, largely within the previously fetched urls, so that the agent set would typically find the already fetched web pages before discovering new ones. This is a manifestation of the exploration vs. exploitation trade-off [Pant et al. 2002]. By exploiting the currently known web graph while cutting down on exploration the system will, eventually, recognize most of the fetched goal pages. When training more slowly, the number of recognized goal pages would usually lag the number of fetched pages by only 2-12 pages. The performance in terms of web fetches improved, at the expense of run time.

3.3 Hayek bid-estimating agents

It seemed clear that a quicker way to hone in on appropriate bid values, particularly during the first 20000 web fetches, would improve the time required to develop an effective set of agents.

The equilibrium bid value approached by bid-ε agents hovers around the point where an agent is profitable and close to break-even. The slow approach of bid-ε agents results in a large amount of exploration as the system cycled through agents. To alleviate this we adopted some ideas from economic theory and equipped each agent with a simple, deterministic intelligence that allowed it to estimate its own "reasonable" bid value, based on immediate rewards, and to refine this bid based on future rewards.

Economic theory [Davis and Holt 1993] suggests that a Nash equilibrium might be a quick way to set up reasonably accurate initial bids. In such a state, bid variations by any one agent decrease its profit. Each agent therefore kept a crude count of its successes and failures, and could estimate its own "worth" and deduce a reasonable bid value (see algorithm 6, pg. 9). The estimates were approximate, especially for short training runs, since future rewards were not immediately added to the self-estimated worth. Although our program provided us with system estimates of the value of links contained in a page, agents usually ignored this information since initial runs showed no large, visible effect of including such information. Presumably this will be useful if better feature sets are implemented for the virtual agents.

In a first-price (pay-what-you-bid) auction the Nash worth of any one bidder is dependent upon the actual bid values of all other agents in the system. The Nash-optimal bid in a second-price auction, however, is to bid exactly the value of the page in question [Davis and Holt 1993]. Such agents need not maintain estimates of how the other agents are bidding. Instead the bid estimate is based only on the estimated immediate reward (goal page or not) and on future returns. For this reason, we adopted a second-price auction mechanism for sets of self-estimating agents.

In our implementation, money from reselling the world was passed to the previous

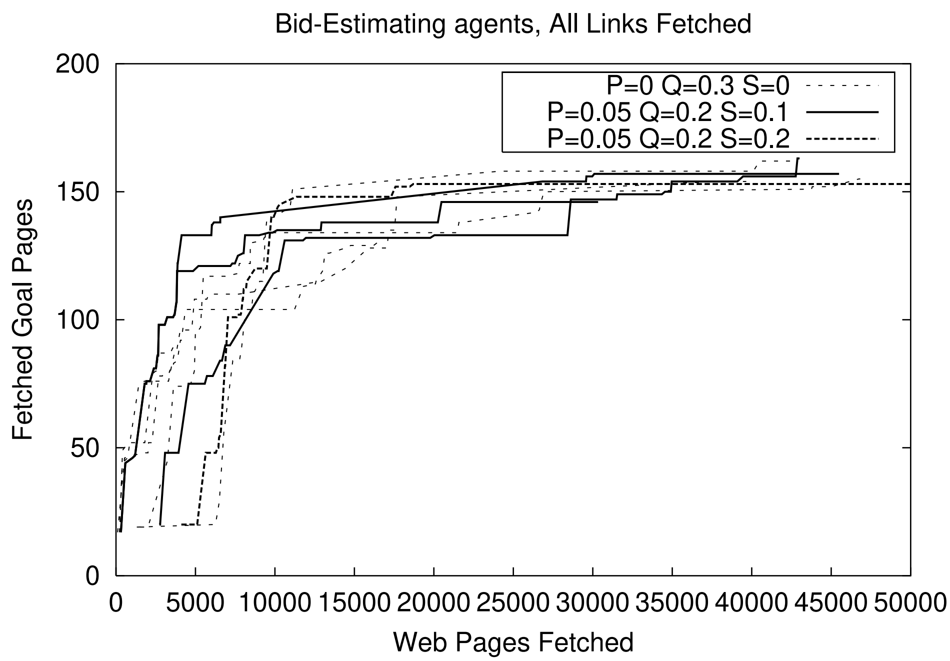


Fig. 4. Tracks show individual training runs using Hayek agents that estimate their own bids. All links were followed. Although virtual documents (and bidders) were allowed, selected urls were almost always chosen based on real web content.

auction winner. This amount is known very quickly and used to adjust the agent's self-estimated bid. Reward kickbacks from future auctions were used as they became known to further refine an agent's self-estimated worth. If the auction terminated before links were traversed, the future Q and S kickbacks (see algorithm 6) effectively had a value of zero, additionally penalizing agents requiring longer paths to goal pages.

Figure 4 shows the number of fetched goal pages using bid-estimating agents when the system is forced to fetch all links. We found that the self-estimating agents did indeed improve quickly and uniformly. These agent sets also were permitted to include agents that bid on virtual documents (i.e. unfetched documents, populated with virtual information such as estimated goal connectivity, text surrounding various links to the url). Although the agents in such a system were allowed to generate and use virtual agents, the fact that all pages in the corona contained real web content meant that the agent sets quickly learned to discriminate against purely virtual agents. However, the system determined that combination agents were important. For example, "volleyball" in a site known to be one-away from goal page derived a higher bid than "volleyball" alone (i.e. not yet known to be one-away).

Using bid-estimating agents (figure 4) and fetching all links, both Q and S-style kickbacks displayed fast convergence to a reasonable bid values. The inconsistent success of the bid-ε agents (figure 3) was much improved.

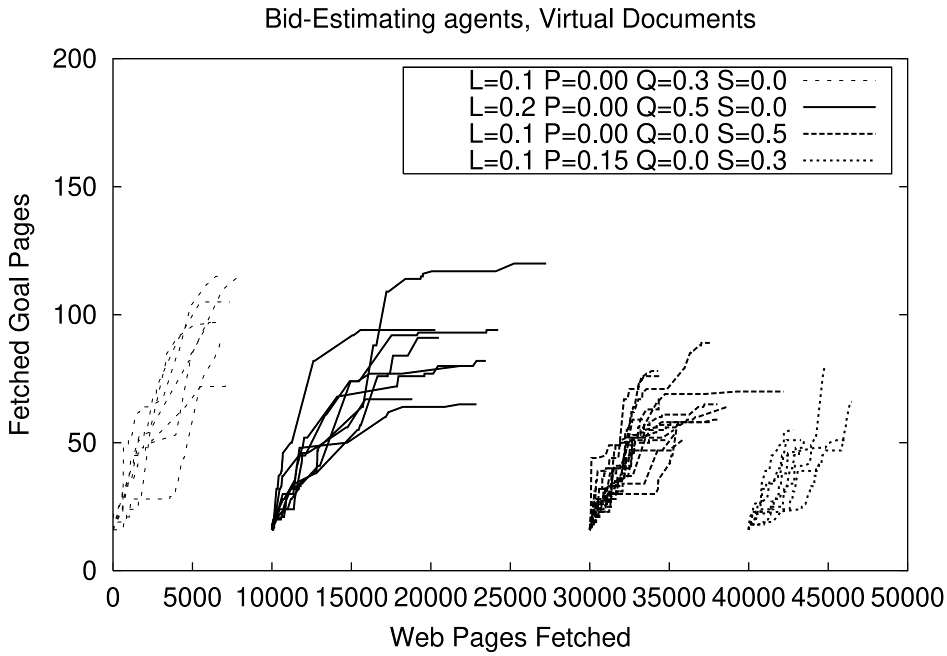


Fig. 5. Tracks show individual training runs using a reasonable [slow] learning rate parameter and a variety of reward-passage schemes. The x-axes have been offset for clarity. The "L" values indicate a fairly slow learning rate, so the lag between fetched and recognized pages was small.

3.4 Learning behavior of bid-estimating agent sets

In this section, we report runs involving virtual agents, which do not automatically fetch page content. Training virtual bidders for the volleyball problem is a harder problem than that tackled by the previous agent sets. Virtual document bidding is expected to be most useful when the system has progressed past the easily found pages into pages of unknown or distant connectivity, or when the initial problem is not well connected (i.e. a set of "competitive" web pages, with few interconnecting links). The fact that the previous fetch-all-links algorithms did as well as they did is related to the fairly high connectivity of the volleyball problem.

In our implementation, only a real goal page has the possibility of generating a goal page reward for its buyer. Virtual bidders that won auctions never collected reward money. Instead they bought a virtual document, retrieved its content and were forced to immediately resell the corona and obtain any added value of the real web content of this page in the form of a (hopefully higher) auction payment. Once a reasonable learning rate was determined, several hundred experiments were run to investigate different profit-passing and bid-estimation methods (figure 5.) The success rate, for this problem, is comparable to that obtained when all links were fetched (figure 4.)

An important factor determining the fast learning in figure 5 seems to be the mechanism used to populate the initial set of agents and to spawn new agents later on. TF, TF-IDF, document count and information gain were available, and in figure 5 information gain criteria [Glover et al. 2002] were usually used to create and spawn agents. The TF-IDF was also a reasonably good mechanism.

We also verified that setting all P, Q and S factors to zero (allowing only market forces to drive the system more slowly toward a local equilibrium) displayed consistently slower learning behavior in the initial training stage. The results were, however, reasonably independent of the actual profit-passing scheme used. It seemed that passing profit to the parent agent (P), for mutation-based spawning of new agents [typically one word would be added (or substituted) from an actual url associated with the spawning agent] was less important in this web context than in other Hayek problems. Presumably this was because information gain or TF-IDF was already a good or better method of agent creation. Passing profit to a previous link owner (S) was effective. About as effective, and simpler to program, was passing profit to the previous auction winner (Q). Historical closeness to goal was also used in the reinforcement learning approach of [Rennie and McCallum 1999].

We note in passing that nonzero S partially simulates running Hayek using a different economic model — namely one where agents buy pages and are compensated whenever another agent buys a page linked from their page. By contrast to the previous Hayek work, where there was only one commodity in the economy — namely the entire world, which was sold in each auction– such an economy has multiple commodities for sale, as occurs in the real economy.

3.5 Comparison of Methods

To facililtate comparison of results, we present again the Bayesian runs and some of the Hayek runs in figure 6. The bid-ε Hayek runs clearly improved upon the Bayesian runs, and the bid-estimating Hayek runs improved again on the bid-ε Hayek runs.

3.6 Performance of trained bid-estimating agent sets

In most applications of focused crawling, one will have an already substantially trained system, and will then apply it. For example, one might have a system trained to find pages on some topic, and use it to maintain one's database as information on the web expands or changes. Thus this section shows that the fully trained agent sets exhibit good crawling behavior if stored and reloaded under different conditions. This allows one to begin with a good set of agents and rapidly advance to difficult problems, continuing the training of a good set of agents by improving the ability to tunnel to goal pages.

Figure 7 shows various restarted runs, using agent sets of varying quality. Within 5000 web fetches, a trained set of agents can do very well, not just fetching goal pages, but also recognizing them and finding previously unfound pages. These runs used virtual documents and agents.

Various stages of training are shown. The highly-trained agent sets were able to recognize 200 goal pages within the first 5000 web fetches, more than obtained in any single training run. In practical terms, this means that an auction-based crawler continues to perform well for the task of maintaining or updating a larger known set of goal pages. Of course, it may be possible to improve these results by adding more classes of features into the agent sets. In this paper, our focus has been on the learning behavior given small starting sets, but these results suggest that a Hayek-style crawler may also be useful for crawls that continually refresh and update a set of already known pages.

We found it better to let the agent set evolve during such restarts and long crawls, rather than to prohibit changes in the bids. The reason was that as more of the web was crawled,

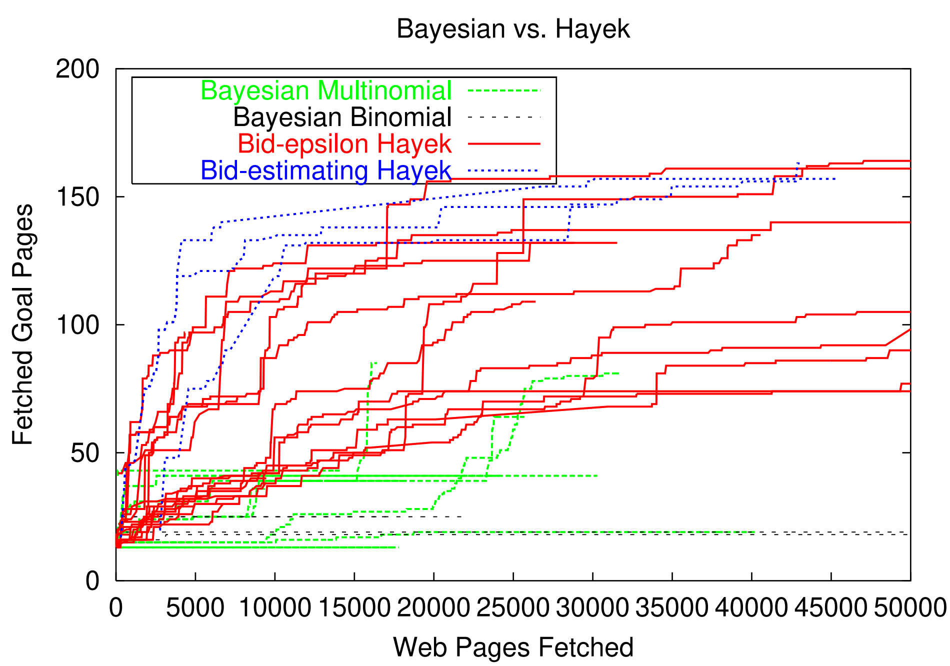


Fig. 6. Comparison of data from other figures. Bayesian monomial and Binomial runs (respectively light dashed lines and sparse dark dashed lines) are copied from figure 2. Bid-ε Hayek runs (solid lines) are copied from figure 3(a). Bid-estimating Hayek runs (dotted lines) are copied from figure 4, but to avoid further clutter we have taken only the P=0.05, Q=0.2, S=0.1 runs from that figure.

distinct web communities would often be found that would unproductively trap any static set of agents and bids, even a supposedly well-trained one. However, repeated failures were noticed and corrected in an evolving Hayek machine. Typical offenders were fresh encounters with generic news sites, e-mail lists, and generic sports sites. A more draconian mechanism was used in the board-of-directors problem to cut short revisiting unproductive sites.

3.7 Board-of-directors pages

We turn now to a very different problem. Searching for company boards of directors pages presents a focused crawler with the more difficult task of learning how to tunnel from one company to another company. The web graph is much less well-connected than the volleyball problem. A simple, points-based arbiter of goal pages was used to define this problem. For this study, the trustworthiness of the goal page judge was not an issue — in fact, perhaps a quarter of the pages were not the actual board page, but something close, and real board of directors pages were sometimes fetched and not recognized by our judge.

An algorithmic change to accommodate the much less connected problem was to blacklist an entire site from the *corona* after a single board of directors page was found. This improvement could still fail: for example, toshiba.com had a board of directors page, but thereafter the system was content to keep crawling amongst different toshiba.com internal sites. A more sophisticated heuristic would improve such cases. Also, for this problem,

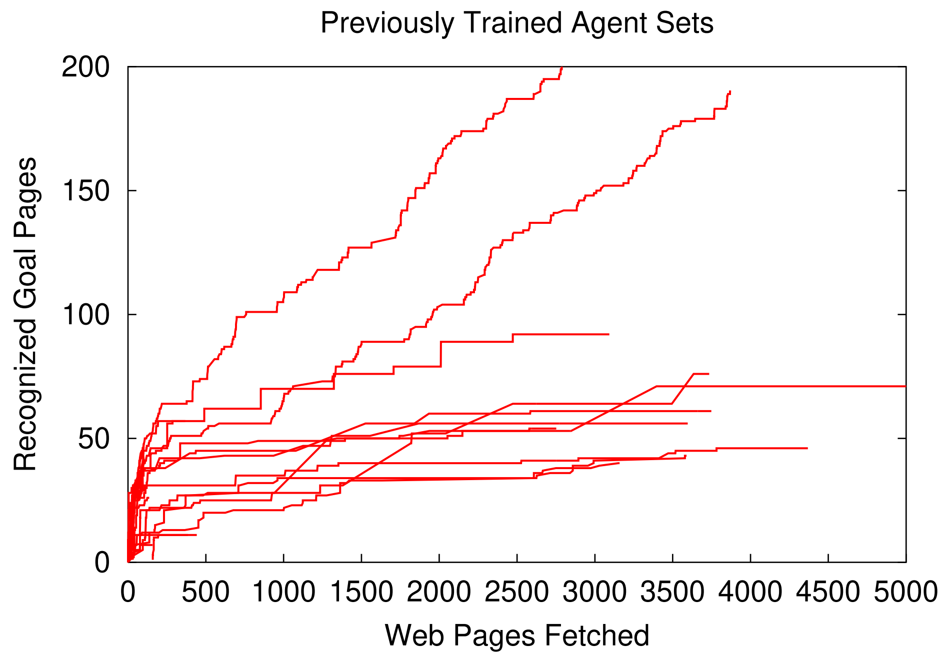


Fig. 7. Tracks of individual crawls show recognized goal pages using various sets of Hayek agents of different quality. Using previously trained agents, it is possible to retrieve large numbers of goal pages quickly, within the first 5000 pages fetched from the web. All links were not automatically fetched: virtual agents were used in these runs.

some pages in the initial Google back-crawl had no link pointing to them. Effectively these pages were worthless for training, since our agents found no links to these pages either.

Other than the above changes, the focused crawl proceeded as for the volleyball problem. Although some experiments were done with the money-passing parameters, we again observed that such variations had little effect on the success rate.

Training that targeted board of directors pages is shown in figure 8. The two groups of runs differ in the rate at which training problem difficulty was increased (parameters in algorithm 2). The slower problem scheduling ended up with starting URLs 2-3 links away from a goal page and did worse than in the volleyball problems of figure 5. The lower connectivity decreases the success rate of the crawl for runs which started under four links away. More unproductive page fetches occur, on average, before the next new goal page is found. The more efficient scheduling was presenting problems beginning 4-5 links away from a goal page when the runs were stopped. The bid-estimating agents did better than a multinomial Bayes crawl that, fetching all links, had selected 41 goal pages out of 91 fetched within the first 12000 page fetches.

We also ran many crawls that began with an initial connected set of 140-200 pages. The pages recognized by agents increased vary rapidly, recognizing the first hundred recognized pages within the first 5000 auctions. Up to 250 goal pages were retrieved in a single restart run without any evidence of saturating the acquisition of new pages.

For this loosely-connected case, one would expect that automatically back-crawling goal

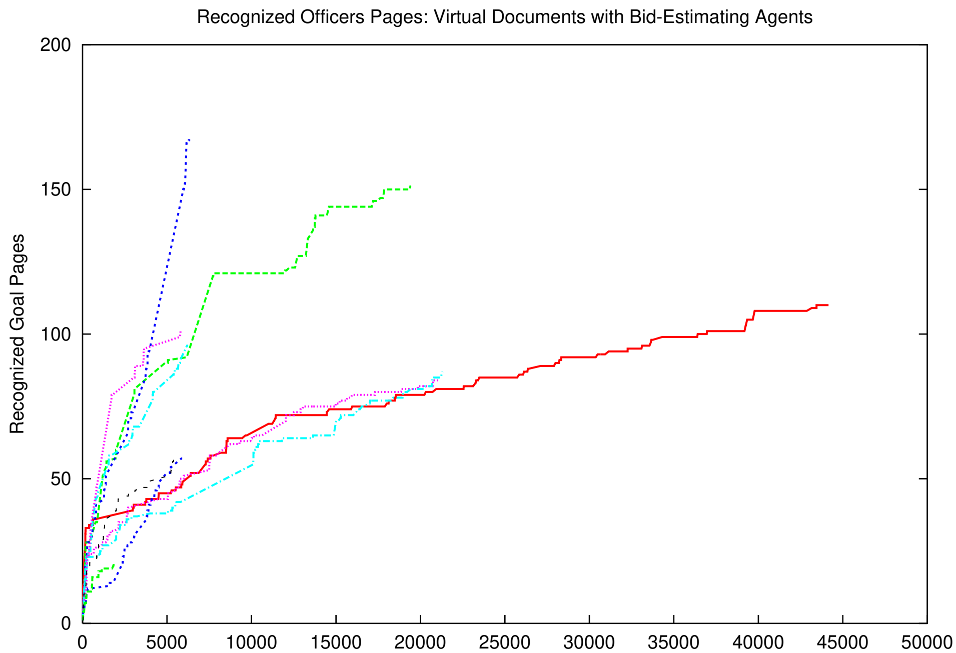


Fig. 8. Tracks of individual training runs show recognized "board of directors" goal pages as a function of the total number of pages fetched. Bid-estimating agents using virtual documents were used. The lower cluster of runs used a different schedule of training from the upper cluster, as described in the text.

pages might be a particularly good way to quickly generate the more diverse training problems that this experiment shows is crucial. Along this vein, relearning the agent set from scratch, but beginning with a more diverse set of problems, was also found to be of possible benefit.

Another practical difference observed was that the runs of this report were done at two sites, one with a T1 connection, and the other a home site with a standard cable connection. For the officer-page problem, the home site did roughly 25% worse, and this was traced to a higher fraction of bad pages. Rather than increase the 20 s page fetch timeout at the home site, we present results from the site with a fast Internet connection (which also ran a very large squid cache used in other crawling research). No such effect was observed for the volleyball problem.

4. FUTURE WORK

Focused crawling using an auction based economy is a flexible mechanism. In spite of considerable success, there is more fundamental work to be done to optimize the performance of these algorithms. The improvements are roughly of three types: improving the overall auction loop, improving the set of agents and features, and improving the economic model. After such improvements have been investigated, it would also be appropriate to address the scalability of the algorithm, as will be addressed below.

In the main auction loop, agent creation mechanisms and more sophisticated problem scheduling were important. Repeated bad sites or bad choices of problem led to the auction sequence being stuck in an unproductive subset of urls. Aborting extensive crawling amongst a very small number of sites was found to be one effective mechanism to promote crawling different sites. A reset mechanism to periodically force the problem difficulty to low values would also be beneficial for the focused crawling problem. Back-crawling, which is a very efficient way to expand a small set of URLs, was only used to set up the initial goal-connected set. Back-crawls whenever any goal page was encountered could easily have been done. The types of "virtual document" information we included could be investigated. Optimization of stopword lists, stemming and hyphenation choices, and stopped URLs were not investigated.

Another set of improvements involves adding more classes of features into the agent sets. These optimizations simply involve allowing a more flexible set of features and virtual features into the agent set: e.g., agents matching word n-grams, Bayesian probability levels of being at various distances from a goal page, ... The auction model itself will weed out the feature types fairly efficiently, as those agents created with the right kinds of features profit and others go broke.

Identifying and increasing visitation of hub sites after analyzing the connectivity of the growing web graph has been found beneficial elsewhere [Pant et al. 2002]. Within the Hayek model, this can be accomplished with virtual features that correspond to several levels of URLs that have been found to be hub sites; however, it would be more productive to generate page features that would allow the agent set itself to learn how to recognize pages likely to be hub sites.

Our simplistic agents using features with only an "And" operator could also easily be made more flexible by allowing "Or", "Near" and "Not" operators. Once virtual bidders of sufficient quality are found, estimated values for links found in pages should be included when setting the *valuation* of bid-estimating agents.

Conceptually, one should realize that the Hayek framework also allows the system to learn heuristic algorithm features. For example, we programmatically stopped visiting a site in the board-of-directors problem after a single goal page was found. A more flexible approach, within the spirit of the economic model is to add bins for the current number of visits or goal pages at a site as virtual features. By representing dynamic crawl state as features, agents can learn to bid, or not bid in given contexts. In this work, we did this when we created agents that mixed goal-connectivity features with word features.

On a more theoretical note, cooperative bidding and profit-sharing are mechanisms that may lead to even more efficient and robust behavior in the economic model. In particular, Fagin's rules [Fagin and Wimmers 2000] provide an attractive mechanism to form subgroups of agents that collectively submit a higher bid for an URL, and share in the proceeds. Other bid-merging economic mechanisms are also possible, as long as one properly addresses the "ownership" issues that arise in developing a collective decision.

After such improvements have been explored, it would be appropriate to experiment with scaling the algorithm to a high bandwidth domain. We envision the current program as useful for home use. As mentioned in section 3.6, one could train this Hayek system to find pages appropriate to some interest of the user, and then run it in background to extract pages not indexed by commercial search engines or to keep current as new pages of interest are added to the web. The results reported on effectiveness of trained Hayek systems indicate that it may already be adequately efficient for this use. Because we had this application in mind, the current system is optimized for use on a relatively low bandwidth, where it has plenty of time for training per (slow) page-fetch. However, it would also be interesting to explore optimizing the Hayek web-crawling program to a high bandwidth environment such as might be faced by a commercial search engine. For such an application, it might be important to speed up the learning by omitting or streamlining some of the more expensive learning calculations.

5. CONCLUSIONS

An economic model can produce an efficient and flexible framework for focused web crawling. The auction economy provides a framework in which a set of agents using widely different classes of feature can together generate an efficient focused crawler. Agents bidding on different feature sets develop and maintain an accurate evaluation of their own worth. Agent sets bidding on combinations of different types of feature learn context-dependent bidding strategies. The learning can be done efficiently even in cases where the initial set of positive examples was very small. For this type of problem, information gain and normalized word count [Weston et al. 2002] scores were found to be good weighting mechanisms for agent generation. Particularly for the difficult case of few positive examples, we found a second-price auction economy that included time-discounted reward/profit back-propagation worked well. Even using only rather trivial agents, we achieved goal page recognition that surpassed the performance of naive Bayesian prioritizers on both the volleyball and board of director problems. Adding Bayesian and feature types that others have determined to be empirically efficient, as well as efficient means to generate plausible agents, can only improve the performance of the Hayek engine used to power a focused crawl. The system works particularly well in what we expect to be its main application: applying a previously trained economy to rapidly find un-indexed pages and keep a database current as new topical pages are added to the web.

Acknowledgement: the authors thank the anonymous referees for many comments that have greatly improved the presentation.

REFERENCES

BAUM, E. B. 1996. Toward a model of mind as a laissez-faire economy of idiots, extended abstract. In *Proc. 13th ICML '96, p28*, L. Saitta, Ed. Morgan Kaufman, San Francisco, CA. and in Machine Learning (1999)v35n2pp155-185.

BAUM, E. B. 1998. Manifesto for an evolutionary economics of intelligence. In *Neural Networks and Machine Learning,p.285*, C. M. Bishop, Ed. Springer-Verlag.

BAUM, E. B. AND DURDANOVIC, I. 2000a. An artificial economy of post production systems. In *Advances in Learning Classifier Systems, Third International Workshop IWLCS 2000*, P. L. Lanzi, W. Stoltzmann, and S. M. Wilson, Eds. Springer, Berlin, 3–21.

BAUM, E. B. AND DURDANOVIC, I. 2000b. Evolution of cooperative problem solving in an artificial economy. *Neural Computation 12,* 12, 2743–2775.

CHAKRABARTI, S., VAN DEN BERF, M., AND DOM, B. 1999. Focused crawling: a new approach to topic-specific web resource discovery. In *Proc. 8th International World Wide Web Conference*. Elsevier Science.

DAVIS, D. D. AND HOLT, C. A. 1993. *Experimental Economics*. Princeton University Press, Princeton, NJ.

DIAO, Y., LU, H., CHEN, S., AND TIAN, Z. 2000. Toward learning based web query processing. In *The VLDB Journal*. 317–328.

DILIGENTI, M., COETZEE, F., LAWRENCE, S., GILES, C. L., AND GORI, M. 2000. Focused crawling using context graphs. In *26th International Conference on Very Large Databases, VLDB 2000*. Cairo, Egypt, 527–534.

ESTER, M., GROSS, M., AND PETER KRIEGEL, H. 2001. Focused web crawling: A generic framework for specifying the user interest and for adaptive crawling strategies. Submitted.

FAGIN, R. AND WIMMERS, E. L. 2000. A formula for incorporating weights into scoring rules. *Theoretical Computer Science 239,* 2, 309–338.

GLOVER, E., TSIOUTSIOULIKLIS, K., LAWRENCE, S., PENNOCK, D., AND FLAKE, G. 2002. Using web structure for classifying and describing web pages. In *International World Wide Web Conference*. Honolulu, Hawaii.

LAWRENCE, S. AND GILES, C. L. 1999. Accessibility of information on the web. *Nature 400,* 6740, 107–109.

LEWIS, D. D. 1992. Representation and learning in information retrieval. Ph.D. thesis, Univ. of Massachusetts.

MCCALLUM, A. AND NIGAM, K. 1998. A comparison of event models for naive bayes text classification. In *Proc. AAAI-98 Workshop on Learning for Text Categorization, 1998.*

MCCALLUM, A., NIGAM, K., RENNIE, J., AND SEYMORE, K. 1999. Building domain-specific search engines with machine learning techniques. In *Proc. AAAI-99 Spring Symposium on Intelligent Agents in Cyberspace, 1999.*

MENCZER, F. AND MONGE, A. E. 1999. Scalable web search by adaptive online agents. In *Intelligent Information Agents*, M. Klusch, Ed. Springer.

NIGAM, K., MCCALLUM, A. K., THRUN, S., AND MITCHELL, T. 2000. Text classification from labeled and unlabeled documents using EM. *Machine Learning 39,* 2/3, 103–134.

PANT, G. AND MENCZER, F. 2002. Myspiders : Evolve your own intelligent web crawlers. *Autonomous Agents and Multi-Agent Systems 5,* 2, 221–229.

PANT, G., SRINIVASAN, P., AND MENCZER, F. 2002. Exploration versus exploitation in topic driven crawlers. In *Proc. WWW-2002 Workshop on Web Dynamics*.

RENNIE, J. AND MCCALLUM, A. K. 1999. Using reinforcement learning to spider the web efficiently. In *Proc. 16th International Conf. on Machine Learning*. Morgan Kaufmann, San Francisco, CA, 335–343.

STEELE, R. 2001. Techniques for specialized search engines. In *Proc. Internet Computing 2001*. Las Vegas.

WESTON, J., PEREZ-CRUZ, F., BOUSQUET, O., CHAPELLE, O., ELISSEEFF, A., AND SCHOELKOPF, B. 2002. Feature selection and transduction for prediction of molecular bioactivity for drug design. In *10th International Conference on Intelligent Systems for Molecular Biology*. Oxford, Edmonton.

Received Month 2002; revised Month Year; accepted Month Year