# Learning to Compete in Heterogeneous Web Search Environments"

Rinat Khoussainov and Nicholas Kushmerick
Department of Computer Science, University College Dublin
Belfield, Dublin 4, Ireland
{rinat, nick}@ucd.ie

## 1 Introduction

Previous research in Web search has mainly targeted performance of search engines from the user's point of view. Parameters such as precision, recall, and freshness of returned results were optimised. On the other hand, a provider of search services is rather interested in parameters like the number of queries processed versus the amount of resources used to process them. We focus on performance optimisation of search engines from the service provider's point of view.

An important factor that affects the search engine performance is competition with other independently controlled search engines. When there are many engines available, users want to send queries to those that provide the best possible results. Thus, the service offered by one search engine influences queries received by others.

Competition is even more important in *heterogeneous search environments* consisting of many specialised search engines (which provide access to the so-called "deep" or "invisible" Web). Each specialised engine in such environment indexes only a subset of all documents (e.g. on a particular topic). Each user query is only sent to a small number of the engines which can return the best results. Selection between specialised search engines is done (semi)automatically by meta-searchers, which rank engines for each user query. Thus, parameters of specialised engines automatically affect their ranking and, hence, the queries they receive, and ultimately their profit.

We are examining the problem of performance-maximising behaviour for non-cooperative specialised search engines in heterogeneous search environments. In particular, we analyse how specialised search engines can select on which topic(s) to specialise and how many documents to index on that topic. We provide a game-theoretic analysis of a simplified version of the problem and motivate the use of the concept of *"bounded rationality".* Bounded rationality assumes that decision makers are unable to act optimally in the game-theoretic sense due to incomplete information about the environment and/or limited computational resources. We then cast our problem as a reinforcement learning task, where the goal of a specialised search engine is to exploit sub-optimal behaviour of its competitors to improve own performance.

*This research was funded by Science Foundation Ireland and the US Office of Naval Research.

## 2 Problem Formalisation

Performance metric. We adopt an economic view on search engine performance. Performance is a difference between the value of the search service provided (income) and the cost of the resources used to provide the service. In our simplified version of the problem we only take into account the cost of resources involved in processing search queries. Under these assumptions, engine performance can be expressed as follows:

$$P = \alpha Q - \beta Q D,$$

where $Q$ is the number of queries received in a given time interval, $D$ is the number of documents in the search engine index, $\alpha$ and $\beta$ are constants.

$\alpha Q$ represents the service value: if the price of processing one search request for a user is $\alpha$, then $\alpha Q$ would be the total income from service provisioning. $\beta Q D$ represents the cost of processing search requests. If $x$ amount of resources is sufficient to process $Q$ queries, then we need $2x$ to process twice as many queries in the same time. Similarly, twice as many resources are needed to search twice as many documents in the same time. Thus, the amount of resources (and, hence, the cost) is proportional to both $Q$ and $D$, and so can be expressed as $\beta Q D$, where $\beta$ reflects the resource costs. The example of the FAST search engine confirms that our cost function is not that far from reality [Risvik and Michelsen, 2002].

Environment model. Let us assume that users issue queries on just a single topic, e.g. "cars". We will see later how this extends to multiple topics. It is reasonable to assume that users would like to send queries to search engines that contain the most relevant documents on the topic, and the more of them, the better. Suppose that search engines have "ideal" Web robots which for a given $D$ can find the $D$ most relevant documents on the topic. In this case, the user would simply have to send queries to the engine containing the largest number of documents on the topic.

This model can be extended to multiple topics, if the state of a search engine is represented by the number of documents $D_i^t$ that engine $i$ indexes for each topic $t$ (a query on topic $t$ would be sent to the engine $i$ with the largest $D_i^t$). Of course, such extension ignores possible overlaps between topics in both queries and documents. On the other hand, if we associate each topic with a search term, the whole engine selection model would closely reflect how some real-life engine selection algorithms work [Callan *et al,* 1995].

Decision making process. For each time interval, the search engines simultaneously and independently decide on how many documents to index and allocate resources to process expected search queries (thus incurring the corresponding costs). As search engines cannot have unlimited crawling resources, we presume that they can only do incremental adjustments to their index contents that require the same time for all engines. The user queries are allocated to the engines based on their index parameters $(D_i^t)$ as described above.

Since a search engine cannot know the number of queries that users will submit, it may allocate less query processing resources than the number of queries it will receive. We assume that excess queries are discarded and the search engine does not benefit from them. Then, performance of engine / over a given time interval can be represented as

$$P_i = \alpha \min(Q_i, \hat{Q}_i) - \beta \hat{Q}_i D_i,$$

where $\hat{Q}_i = \sum_{t:D_i^t>0} \hat{Q}_i^t$ is the expected number of queries across all topics for which the search engine is trying to compete (i.e. index at least one d o c nt), $D_i = \sum_t D_i^t$ s the total number of indexed documents, $Q_i = \sum_t Q_i^t$ is the total number of queries actually received by engine *i,* and

$$Q_i^t = \begin{cases} 0 & : \quad \exists j, D_i^t < D_j^t \\ \frac{Q^t}{|K|} & : \quad i \in K, K = \{k : D_k^t = \max_j D_j^t\} \end{cases}$$

is the share of queries in topic / received by engine i, and $Q^t$ is the number of queries users submitted on topic *t*.

## 3 Analysis and Solution Approach

The decision-making process can be modelled as a multi-stage game. At each stage, a matrix game is played, where players are search engines, actions are values of *$(D^t)$,* and player *i* receives payoff $P_i$ (as above). If player ,• knew the actions of its opponents at a future stage k, it could calculate the optimal response as the one maximising its payoff $P_i(k)$ at that stage. For example, in case of a single topic and a constant query stream it should play $D_i(k) = \max_{j \neq i} D_j(k) + 1$, if $\max_{j \neq i} D_j(k) + 1 < \alpha/\beta$, and $D_i(k) = 0$ otherwise (simply put, outperform opponents by 1 document if profitable, and do not incur any costs otherwise).

In reality, the players do not know the future. One possible way around this would be to agree on (supposedly, mutually beneficial) actions in advance. To avoid deception, players would have to agree on playing a Nash equilibrium of the game, since only then there will be no incentive for them to not follow the agreement. Agreeing to play a Nash equilibrium, however, becomes problematic when the game has multiple such equilibria. Players would be willing to agree on a Nash equilibrium yielding to them the highest (expected) payoffs, but the task of characterising all Nash equilibria of a game is NP-hard even given complete information about the game (as follows from [Conitzer and Sandholm, 2002]).

NP-hardness results and the possibility that players may not have complete information about the game lead to the idea of "bounded rationality", when players may not use the optimal strategies in the game-theoretic sense. Our proposal is to cast the problem of optimal behaviour in the game as a learning task, where the player would have to learn a strategy that performs well against its sub-optimal opponents.

Learning in repeated games has been studied extensively in game theory and machine learning. Examples include fictious play and opponent modelling [Robinson, 1951; Carmel and Markovitch, 1996]. We apply a more recent algorithm from reinforcement learning called GAPS [Peshkin *et al,* 2000]. In GAPS, the learner plays a parameterised strategy represented by a finite state automaton, where the parameters are the probabilities of actions and state transitions. GAPS implements stochastic gradient ascent in the space of policy parameters. After each learning trial, parameters of the policy are updated by following the payoff gradient.

GAPS has a number of advantages important for our domain. It works in partially observable games. It also scales well to multiple topics by modelling decision-making as a game with factored actions (where action components correspond to topics). The action space in such games is the product of factor spaces for each action component. GAPS, however, allows us to reduce the learning complexity: rather than learning in the product action space, separate GAPS learners can be used for each action component. It has been shown that such distributed learning is equivalent to learning in the product action space. We call a search engine that uses the proposed approach COUGAR, which stands for Competitor Using GAPS Against Rivals.

## 4 Preliminary Results

We evaluated our approach in a number of simulation experiments, which demonstrated COUGAR's ability to compete successfully with different opponents. For the sake of brevity, we present here results from one such experiment with two competing search engines. One search engine was using a fixed strategy, called "Bubble", the other one was COUGAR. The search engines could increase, decrease (by one), or keep the number of documents indexed on each topic. They could also observe the size of own index, the relative sizes of opponents' indices, and the number of user queries $Q^t$ submitted for each topic (the last two observations can be obtained from the meta-searcher). The expected number of queries on topic *t* at future stage A; was calculated as $\hat{Q}_i^t(k) = Q^t(k-1)$.

The experimental setup consisted of three components: the generator of search queries, the metasearcher, and the search engines. The state of a search engine's document index is represented by a vector $(D_i^t)$ of the numbers of documents indexed by the search engine for each topic. This is the information used by the metasearcher to select search engines.

To simulate user search queries, we used HTTP logs obtained from a Web proxy of a large ISP. We developed extraction rules individually for 47 well-known search engines. The total number of queries extracted was 657,861 collected over a period of 190 days. We associated topics with search terms in the logs. To simulate queries for *T* topics, we extracted the *T* most popular terms from the logs. The number of queries generated on topic *t* for a given day was equal to the number of queries with term *t* in the logs belonging to this day.

The "Bubble" strategy tries to index as many documents as possible without any regard to what competitors are do-

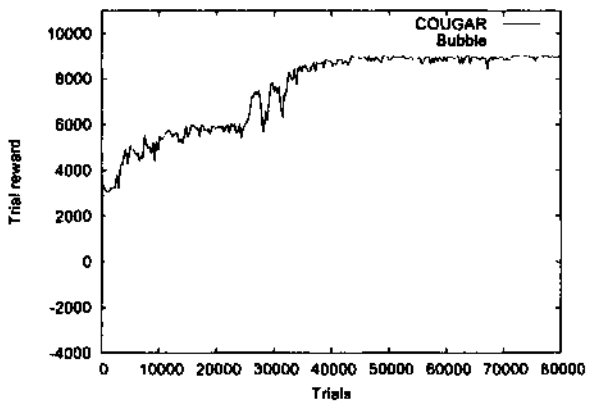

Figure 1: "Bubble" vs COUGAR: learning curve

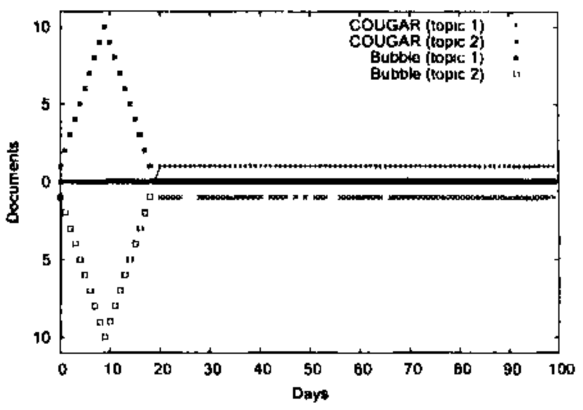

Figure 2: "Bubble" vs COUGAR: sample trial

ing. As follows from our performance formula (Section 2), such unconstrained growing leads eventually to negative performance. Once the total reward falls below a certain threshold, the "Bubble" search engine goes bankrupt (it shrinks its index to 0 documents and retires until the end of the trial). This process imitates the situation, in which a search provider expands its business without paying attention to costs, and eventually runs out of money (or an analogy with the " . com bubble"). An intuitively sensible response to the "Bubble" strategy would be to wait until the bubble "bursts" and then come into the game alone. That is, a competitor should not index anything while the "Bubble" grows and should start indexing a minimal number of documents once the "Bubble" search engine goes bankrupt.

Our simulation had two topics, and "Bubble" was increasing (and decreasing) the number of documents indexed for each topic simultaneously. First, we trained COUGAR in a number of learning trials. Once it reached a steady performance level, the resulting strategy was evaluated in a scries of testing trials. Each simulation trial consists of 100 days, where each day corresponds to one stage of the multi-stage game played. The resulting performance in the whole trial is calculated as a sum of discounted rewards from each day.

Figure 1 shows how COUGAR's performance improving during learning. Figure 2 visualises a testing trial between the "Bubble" and the COUGAR engines by showing the number of documents indexed by the engines on each day of the trial (the top half of Y axis shows the number of documents for topic 1, the bottom half shows the number of documents for topic 2). Note that COUGAR has learned to wait until "Bubble" goes bankrupt, and then to win all queries for both topics.

We also investigated effectiveness of our approach against evolving opponents. In particular, we evaluated performance of a COUGAR strategy learned in self-play against other learners (other COUGARs in our case). We observed that while COUGAR's performance can be quite stable against equally complex learners, it may still be sub-optimal, in the sense that a "cleverer" learner (e.g. a COUGAR with more policy states) can outperform it.

## 5 Future Work

We do not claim to provide a complete solution for the problem of performance management in heterogeneous Web search, but COUGAR is a promising first step. Clearly, we have made many strong assumptions in our models. One future direction will be to relax these assumptions to make our simulations more realistic. Another important direction would be to further study performance of the learning algorithm in self-play, as well as against other learners.

While we are motivated by the optimal behaviour for search services over document collections, our approach is applicable in more general scenarios involving services that must weigh the cost of their inventory of objects against the expected inventories of their competitors and the anticipated needs of their customers. For example, it would be interesting to apply our ideas to large retail e-commerce sites which must decide what products to stock.

## References

[Callan *et al.,* 1995] J. P. Callan, Z. Lu, and W. B. Croft. Searching distributed collections with inference networks. In *Proc. of the 18th Annual Intl. ACM S1G1R Conf,* pages 21-28. ACM Press, July 1995.

[Carmel and Markovitch, 1996] D. Carmel and S. Markovitch. Learning models of intelligent agents. In *Proc. of the 13th National Conf. on Artifical Intelligence,* pages 62-67, Menlo Park, CA, 1996.

[Conitzer and Sandholm, 2002] V. Conitzer and T. Sandholm. Complexity results about Nash equilibria. Technical Report CMU-CS-02-135, Camegie Mellon University, 2002.

[Peshkin etal.,2000] L. Pcshkin, N. Meuleau, K.-E. Kim, and L.Kaelbling. Learning to cooperate via policy search. In *Proc. of the 16th Conf on Uncertaintly in Artifical Intelligence,* pages 489-496. Morgan Kaufmann, 2000.

[Risvik and Michelsen, 2002] K. Risvik and R. Michelsen. Search engines and web dynamics. *Computer Networks,* 39:289-302, June 2002.

[Robinson, 1951] J. Robinson. An iterative method of solving a game. *Annals of Mathematics,* 54:296-301, 1951.