Controlling the Robots of Web Search Engines

J. Talim
Department of Mathematics and Statistics
University of Saskatchewan
Saskatchewan, Saskatoon
S7N 5E6 Canada
talim@snoopy.usask.ca

Z. Liu
IBM Research
Hawthorne, NY 10532
zhenl@us.ibm.com

Ph. Nain
INRIA
Sophia Antipolis
06902 France
nain@sophia.inria.fr

E. G. Coffman, Jr.
Electrical Engineering Dept.
Columbia University
New York, NY 10027
egc@ee.columbia.edu

Published in: *Proc. ACM Sigmetrics 2001 / Performance 2001 Conf. Performance Evaluation Review*, Vol. 29, No. 1, pp. 236-244, June 2001

Abstract

Robots are deployed by a Web search engine for collecting information from different Web servers in order to maintain the currency of its data base of Web pages. In this paper, we investigate the number of robots to be used by a search engine so as to maximize the currency of the data base without putting an unnecessary load on the network. We adopt a finite-buffer queueing model to represent the system. The arrivals to the queueing system are Web pages brought by the robots; service corresponds to the indexing of these pages. Good performance requires that the number of robots, and thus the arrival rate of the queueing system, be chosen so that the indexing queue is rarely starved or saturated. Thus, we formulate a multi-criteria stochastic optimization problem with the loss rate and empty-buffer probability being the criteria. We take the common approach of reducing the problem to one with a single objective that is a linear function of the given criteria. Both static and dynamic policies can be considered. In the static setting the number of robots is held fixed; in the dynamic setting robots may be re-activated/de-activated as a function of the state. Under the assumption that arrivals form a Poisson process and that service times are independent and exponentially distributed random variables, we determine an optimal decision rule for the dynamic setting, i.e., a rule that varies the number of robots in such a way as to minimize a given linear function of the loss rate and empty-buffer probability. Our results are compared with known results for the static case. A numerical study indicates that substantial gains can be achieved by dynamically controlling the activity of the robots.

Keywords: Web search engines; Web robots; Queues; Markov decision process.

1 Introduction

The World Wide Web has become a major information publishing and retrieving mechanism on the Internet. The amount of information as well as the number of Web servers continues to grow exponentially fast. In order to help users find useful information on the Web, search engines such as Alta Vista, HotBot, Yahoo, Google, Infoseek, Magellan, Excite and Lycos, etc. are available. These systems consist of four main components: a database that contains web pages (full text or summary), a user interface that deals with queries, an indexing engine that updates the database, and robots that traverse the Web servers and bring Web pages to the indexing engine. Thus, the quality of a search engine depends on many factors, e.g., query response time, completeness, indexing speed, currency, and efficient robot scheduling.

Our interest here focuses on the function performed by robots: establishing currency by bringing new pages to be indexed and bringing changed/updated pages for re-indexing. We investigate the problem of choosing the number of robots to meet the conflicting demands of low network traffic and an up-to-date data base. The specific model, illustrated in Figure 1, centers on the indexing engine, which is represented by a single-server queue with finite buffer, and multiple robots acting as sources of arriving pages. The times between successive page accesses are independent and identically distributed for each robot; the robots themselves are identical and function independently. The indexing (service) times are independent, identically distributed, and independent of the arrival processes.

When a robot arriving with a page finds the indexing buffer full, the page being delivered is lost, at least temporarily. In this situation, a potential update or new page has been lost, and network congestion has been created to no benefit. On the other hand, if the buffer is ever empty, and hence the indexing engine is idle, data base updating is at a standstill waiting for the robots to bring more pages. To reduce the probability of the first of these two events, we want to keep the number of robots suitably small, but to reduce the probability of the second, we want to keep the number of robots suitably large. To make the objective concrete, we will formulate a cost function as a weighted sum of the loss rate and the probability of an empty buffer. We will then study the problem of varying the number of robots in a way that minimizes this cost function.

In a more general set-up, the parameters of robot scheduling problems like ours might represent explicit limits not only on buffer/storage space, but also on network bandwidth and the obsolescence of stored documents. The first two constraints combine in certain cases in that losses created by bandwidth limits are modeled by losses created by bounds on buffer capacity. If the occupancy of the indexing buffer is frequently too large, then during many of the waiting times, buffered documents are modified at their home sites, thus making them obsolete before they are even made a part of the data base. It is thus useful to bound waiting times by requiring (e.g.) a small buffer.

Because of the obsolescence issue, we would have to bound the waiting time either deterministically or stochastically, even when the buffer size is effectively infinite. The objective function would be slightly modified. Such variants are certainly worthwhile analyzing, and we propose them here as directions for further research. In this paper, the focus is on the simpler finite-buffer model, which is still rich enough to serve as a useful reference model for studying the balance between network congestion and server utilization.

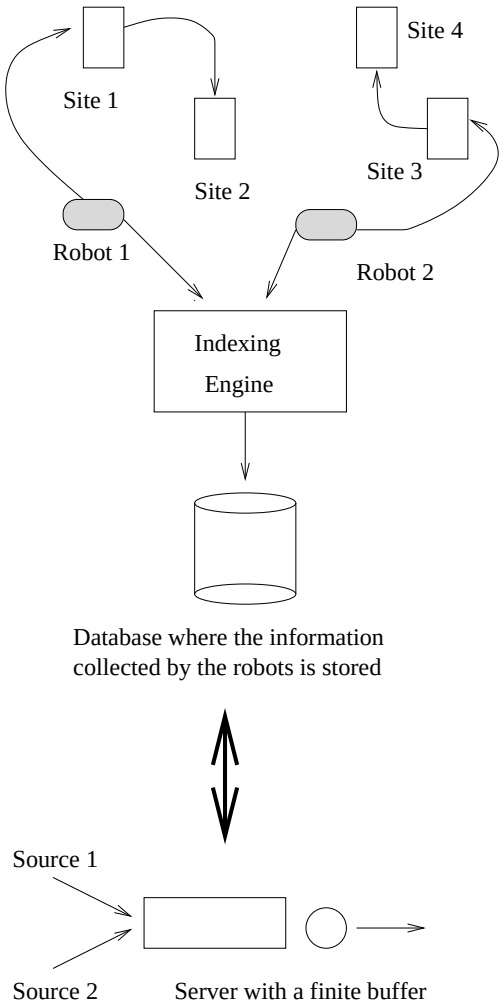

Figure 1: Model of search engine with 2 robots

There is a large literature on search engines and their components. The search engines themselves may well be their own best source of references; we recommend this entree to the research on any aspect of the subject. In particular, much can be found on the design and control, including distributed control, of robots. However, there appears to be very little on the modeling and analysis of robot scheduling and the indexing queue. The early work in [2] proposes a natural model of Web-page obsolescence, and studies the problem of scheduling a single search engine robot so as to minimize the extent to which the search engine's data base is out-of-date. Our work in [6] introduces the multiple-robot model of this paper, but studies the static optimization problem where the number of robots stays fixed.

After the preliminaries of the next section, we present our results on the dynamic robot scheduling problem in Section 3. The tools that we use are taken from the theory of Markov decision processes. Numerical results comparing static and dynamic policies are reported in Section 4. Section 5 concludes with a brief discussion of promising open issues.

2 Preliminaries

We assume that the number of active robots may vary in time as a function of the backlog in the queue and the number of robots already active. To address this situation we will cast our model into the Markov Decision Process (MDP) framework [1, 4, 5]. First, we need to define the queueing model.

The indexing engine is modeled as a finite-capacity single-server queue. Service times are i.i.d. exponentially distributed random variables with mean $1/\mu$ and the buffer may accomodate at most K customers, including the one in service, if any, where $K > 2$. There are N available robots and each of these robots, when activated, brings pages to the server in a Poisson stream at rate λ. We assume that these N Poisson processes are mutually independent and independent of the service times.

The novel feature of our model is that the number of active robots may be modified at any arrival or departure epoch. When an arrival occurs, the incoming robot is de-activated at once; the controller may then decide to keep it idle or to re-activate it. When a departure occurs the controller may either decide to activate one additional robot, if one is available, or to do nothing (i.e. the number of active robots is left unchanged). The objective is to find a policy (to be defined) that minimizes a weighted sum of the stationary starvation probability and loss rate.

We now introduce the MDP setting in which we will solve this optimization problem. Since the time between transitions is variable we will use the uniformization method [1, Sec. 6.7].

At the n-th decision epoch t_n the state of the MDP is represented by the triple $x_n = (q_n, r_n, s_n) \in \{0, 1, \ldots, K\} \times \{0, 1, \ldots, N\} \times \{0, 1, 2\}$, with q_n and r_n the queue-length and the number of active robots just *before* the n-th decision epoch, respectively, and s_n the type (arrival, departure, fictitious – see below) of the n-th decision epoch.

The successive decision epochs $\{t_n, n \geq 1\}$ are the jump times of a Poisson process with intensity $\nu := \lambda N + \mu$, independent of the service time process. In this setting, the n-th decision epoch t_n corresponds to an arrival in the original system with probability $\lambda r_n/\nu$ (in which case $s_n = 1$), to a departure with probability μ/ν provided that $q_n > 0$ ($s_n = 0$), and to a fictitious event with the complementary probability $((N - r_n)\lambda + \mu)/\nu$ ($s_n = 2$).

Let $a_n \in \{0, 1\}$ be the action chosen at time t_n. We assume that $a_n = 1$ if the decision is made to activate one additional robot, if one is available, and $a_n = 0$ if the decision is made to keep the number of active robots unchanged. By convention we assume that $a_n = 0$ if the n-th decision epoch corresponds to a fictitious event ($s_n = 2$).

From the above definitions we see that states of the form $(\bullet, 0, 1)$ and $(0, \bullet, 0)$ are not feasible, as an arrival cannot occur if all robots are inactive and a departure cannot occur if the queue is empty, respectively. Therefore, the state-space for this MDP is

$$\{(q, r, s), 0 \leq q \leq K, 0 \leq r \leq N, s = 0, 1, 2\} - \{(0, r, 0), (q, 0, 1), 0 \leq q \leq K, 0 \leq r \leq N\}.$$

However, this set contains one absorbing state, the "fictitious" state $(0, 0, 2)$. To remove this undesirable state we will only consider policies (see formal definition below) that always choose action

$a = 1$ when the system is in state $(1, 0, 0)$ so that $(0, 0, 2)$ can never be reached. This is not a severe restriction since a policy that never activates robots when the system is empty is of no interest. In conclusion, the state space for this MDP is

$$\begin{aligned}\mathbf{X} := &\{(q, r, s), 0 \leq q \leq K, 0 \leq r \leq N, s = 0, 1, 2\} \\ &-\{(0, 0, 2), (0, r, 0), (q, 0, 1), 0 \leq q \leq K, 0 \leq r \leq N\}\end{aligned}$$

and the set $\mathbf{A}_x$ of allowed actions when the system is in state $x = (q, r, s) \in \mathbf{X}$ is given by

$$\mathbf{A}_x = \begin{cases} \{0\} & \text{if } s = 2 \\ \{1\} & \text{if } (q, r, s) = (1, 0, 0) \\ \{0, 1\} & \text{otherwise.} \end{cases}$$

To complete the definition of the MDP we need to introduce the one-step cost c and the one-step transition probabilities p. Given that the process is in state $x = (q, r, s)$ and that action a is made, the one-step cost is defined as

$$c(x) = \gamma\, \mathbf{1}(q = 0) + \nu\, \mathbf{1}(q = K, s = 1), \tag{1}$$

independent of a. We will show later that this choice for the one-step cost will allow us to address, and subsequently to solve, the optimization problem at hand.

For $x \in \mathbf{X}$, the one-step transition probabilities $p_{x,x'}(a)$ are given by

$$p_{x,x'}(a) = \begin{cases} \dfrac{\mu}{\nu}\, \mathbf{1}(q > 1) & \text{if } x' = (q - 1, \min\{r + a, N\}, 0) \\ \dfrac{\lambda r}{\nu} & \text{if } x' = (q - 1, \min\{r + a, N\}, 1) \\ 1 - \dfrac{\mu\, \mathbf{1}(q > 1) - \lambda r}{\nu} & \text{if } x' = (q - 1, \min\{r + a, N\}, 2) \end{cases} \tag{2}$$

if $s = 0$, $a = 0, 1$;

$$p_{x,x'}(a) = \begin{cases} \dfrac{\mu}{\nu} & \text{if } x' = (\min\{q + 1, K\}, r + a - 1, 0) \\ \dfrac{\lambda(r + a - 1)}{\nu} & \text{if } x' = (\min\{q + 1, K\}, r + a - 1, 1) \\ 1 - \dfrac{\mu + \lambda(r + a - 1)}{\nu} & \text{if } x' = (\min\{q + 1, N\}, r + a - 1, 2) \end{cases} \tag{3}$$

if $s = 1$, $a = 0, 1$;

$$p_{x,x'}(0) = \begin{cases} \dfrac{\mu}{\nu}\, \mathbf{1}(q > 0) & \text{if } x' = (q, r, 0) \\ \dfrac{\lambda r}{\nu} & \text{if } x' = (q, r, 1) \\ 1 - \dfrac{\mu\, \mathbf{1}(q > 0) + \lambda r}{\nu} & \text{if } x' = (q, r, 2) \end{cases} \tag{4}$$

if $s = 2$. All other transition probabilities are equal to 0.

Without loss of generality we will only consider *pure stationary* policies since it is known that nothing can be gained by considering more general policies [4, Ch. 8-9]. Recall that in the MDP setting a policy π is pure stationary if, at any decision epoch, the action chosen is a non-randomized and time-homogeneous mapping of the current state [1, 4, 5]. We define an *admissible* stationary policy as any mapping $\pi : \mathbf{X} \to \{0, 1\}$ such that $\pi(x) \in \mathbf{A}_x$.

For later use introduce $P(\pi) := [p_{x,x'}(\pi(x))]_{(x,x') \in \mathbf{X} \times \mathbf{X}}$, the transition probability matrix under the stationary policy π.

Let $\mathcal{P}$ be the class of all admissible stationary policies. For any policy $\pi \in \mathcal{P}$ introduce the long-run expected average cost per unit time

$$W_\pi(x) = \lim_{n \to \infty} \frac{1}{n} \mathbf{E}_\pi \left[\sum_{i=1}^{n} c(x_i) \,|\, x_1 = x \right], \qquad x \in \mathbf{X}. \tag{5}$$

The existence of the limit in (5) is a consequence of the fact that π is stationary and $\mathbf{X}$ is countable [4, Proposition 8.1.1]. We shall say that a policy $\pi^\star \in \mathcal{P}$ is average cost optimal if

$$W_{\pi^\star}(x) = \inf_{\pi \in \mathcal{P}} W_\pi(x) \qquad \forall x \in \mathbf{X}. \tag{6}$$

3 Results

In order to use results from MDP theory for average cost models, we first need to determine the class (recurrent, unichain, multichain, communicating, etc.) in which the current MDP belongs. Consider the following example: Let $N = 2$ and let π be any stationary policy that selects action 1 in states $(\bullet, r, 1)$ for $r \in \{1, 2\}$ and in state $(1, 0, 0)$, and action 0 otherwise. It is easily seen that this policy induces an MDP with two recurrent classes ($\mathbf{X} \cap \{(\bullet, 1, \bullet)\}$ and $\mathbf{X} \cap \{(\bullet, 2, \bullet)\}$ and a set of transient states ($\mathbf{X} \cap \{\bullet, 0, \bullet\}$). We therefore conclude from this example that the MDP $\{x_n, n \geq 1\}$ is *multichain* [4, p. 348].

An MDP is *communicating* [4, p. 348] if, for every pair of states $(x, x') \in \mathbf{X} \times \mathbf{X}$, there exists a stationary policy π such that x' is accessible from x, that is, if there exists $n \geq 1$ such that $P^n_{x,x'}(\pi) > 0$, where $P^n_{x,x'}(\pi)$ is the (x, x')-entry of the matrix $P^n(\pi)$.

Lemma 1 *The MDP $(x_n, n \geq 1)$ is communicating.* ◇

The proof of Lemma 1 can be found in the Appendix. The next result flows from Lemma 1 and Proposition 4 in [1, Sec. 7.1]:

Proposition 1 *There exists a scalar θ and a mapping $h : \mathbf{X} \to \mathbb{R}$ such that, for all $x \in \mathbf{X}$,*

$$\theta + h(x) = c(x) + \min_{a \in \mathbf{A}_x} \sum_{x' \in \mathbf{X}} p_{x,x'}(a)\, h(x') \tag{7}$$

with $\theta = \inf_{\pi \in \mathcal{P}} W_\pi(x)$ *for all* $x \in \mathbf{X}$*, while if* $\pi^\star(x)$ *attains the minimum in (7) for each* $x \in \mathbf{X}$*, then the stationary policy* $\pi^\star$ *is optimal.* $\diamond$

The optimal average cost θ and the optimal policy $\pi^\star$ in Proposition 5 can be computed by using the following recursive scheme, known as the relative value iteration algorithm.

Proposition 2 *Let* $\hat{x}$ *be a fixed state in* $\mathbf{X}$ *and let* τ*,* $0 < \tau < 1$ *be a fixed number. For* $k \geq 0$*,* $x \in \mathbf{X}$*, define the mappings* $(h_k,\ k \geq 0)$ *as*

$$h_{k+1}(x) = (1-\tau)h_k(x) + \tau\left(T(h_k)(x) - T(h_k)(\hat{x})\right)$$

with

$$T(h_k)(x) := c(x) + \min_{a \in \mathbf{A}_x} \sum_{x' \in \mathbf{X}} p_{x,x'}(a)\, h_k(x'),$$

where $h_0(\hat{x}) = 0$ *but otherwise* h_0 *is arbitrary.*

Then, the limit $h(x) = \lim_{k\to\infty} h_k(x)$ *exists for each* $x \in \mathbf{X}$*,* $\theta = \tau T(h)(\hat{x})$*, and the optimal action* $\pi^\star(x)$ *in state* x *is given by* $\pi^\star(x) \in \operatorname{argmin}_{a \in \mathbf{A}_x} \sum_{x' \in \mathbf{X}} p_{x,x'}(a)\, h(x')$*.* $\diamond$

Proof. Since the MDP is communicating (cf. Lemma 1) the proof follows from [4, Sec. 8.5,9.5.3] (see also [1, Prop. 4, p. 313]). ∎

We now return to our initial objective, namely, minimizing a weighted sum of the stationary starvation probability and loss rate. To see why the solution to this problem is given by the solution to the MDP problem formulated earlier, it suffices to show that the average cost (5) is a weighted sum of the stationary starvation probability and loss rate. It should be clear, however, that this result cannot hold for policies that induce an average cost (5) that depends on the initial state x since, by definition, the stationary starvation probability and loss rate are independent of the initial state. We will therefore restrict ourselves to the class $\mathcal{P}_0 \subset \mathcal{P}$ of policies that generate a constant average cost, namely, $\mathcal{P}_0 = \{\pi \in \mathcal{P} : W_\pi(x) = W_\pi(x'), \forall x \in \mathbf{X}\}$.

The set $\mathcal{P}_0$ is non-empty as it is well known to contain, among other policies, all unichain policies [4, Proposition 8.2.1]. Among such policies is the *static* policy π_N that always maintains N robots active, namely, $\pi_N(x) = 1$ for all $x = (\bullet, \bullet, s) \in \mathbf{X}$ with $s = 0, 1$ and $\pi_N(x) = 0$ for all $x = (\bullet, \bullet, 2) \in \mathbf{X}$.

Note also that reducing the search for an optimal policy to policies in $\mathcal{P}_0$ does not yield any loss of generality as it is also known that there always exits an optimal policy with constant average cost in the case of communicating MDP's [4, Proposition 8.3.2].

Fix $\pi \in \mathcal{P}_0$. Introducing (1) into (5) yields $W_\pi(x) = \gamma S_\pi(x) + L_\pi(x)$ with

$$\begin{aligned} S_\pi(x) &= \lim_{n\to\infty} \frac{1}{n}\, \mathbf{E}_\pi \left[\sum_{i=1}^{n} \mathbf{1}(q_i = 0) \,|\, x_1 = x\right] \\ L_\pi(x) &= \nu \lim_{n\to\infty} \frac{1}{n}\, \mathbf{E}_\pi \left[\sum_{i=1}^{n} \mathbf{1}(q_i = K, s_i = 1) \,|\, x_1 = x\right]. \end{aligned}$$

In the following we will drop the argument x in $S_\pi(x)$ and $L_\pi(x)$ since these quantities do not depend on x from the definition of $\mathcal{P}_0$.

Let us now interpret S_π and L_π. S_π is the stationary probability that the system is empty at decision epochs. Since the decision epochs form a Poisson process, we may conclude from the PASTA property [7] that S_π is also equal to the stationary probability that the system is empty at *arbitrary epoch*, and this in nothing more than the stationary starvation probability.

Let us now consider L_π. Recall that $\{t_n, n \geq 1\}$, the sequence of decision instants, is a Poisson process with intensity ν and assume without loss of generality that $t_1 = 0$ a.s. Define $A(t)$ as the total number of customers that have arrived to the queue up to time t, including customers which have been lost, and let $Q(t)$ be the queue length at time t. We assume that the sample paths of the processes $\{A(t),\, t \geq 0\}$ and $\{Q(t),\, t \geq 0\}$ are right-continuous with left limits. With these definitions and the identity $\mathbf{E}[t_n] = n/\nu$ we may rewrite L_π as

$$L_\pi = \lim_{n\to\infty} \frac{\mathbf{E}_\pi \left[\int_0^{t_n} \mathbf{1}(Q(t-) = K)\, dA(t)\right]}{\mathbf{E}[t_n]}.$$

In other words, we have shown that L_π is the ratio, as n tends to infinity, of the expected number of losses during the first n decision epochs over the expected occurence time of the n-th decision epoch.

The interpretation of L_π as a *loss rate* now follows from the identity

$$L_\pi = \lim_{n\to\infty} \frac{\mathbf{E}_\pi \left[\int_0^{t_n} \mathbf{1}(Q(t-) = K)\, dA(t)\right]}{\mathbf{E}[t_n]} = \lim_{T\to\infty} \frac{1}{T}\, \mathbf{E}_\pi \left[\int_0^{T} \mathbf{1}(Q(t-) = K)\, dA(t)\right], \quad \forall \pi \in \mathcal{P}_0, \tag{8}$$

upon noticing that the latter quantity represents the mean number of losses per unit time or the loss rate. The second identity in (8) is a direct consequence of the theory of renewal reward processes [5, Theorem 7.5] and of the definition of the set $\mathcal{P}_0$.

In summary, we have shown that for any policy π in $\mathcal{P}_0$ the average cost is

$$W_\pi = \gamma S_\pi + L_\pi,$$

with S_π the starvation probability and L_π the loss rate. ∎

The optimal policy has been computed for different values of the model parameters. Figures 2-4 display the optimal policy for $N = 16$, $K = 5$, $\lambda = 0.1$, $\mu = 1.0$ and for different values of γ ($\gamma < \gamma(K) = 1.4$, $\gamma = \gamma(K)$ and $\gamma > \gamma(K)$). The results were obtained by running the value iteration algorithm given in Proposition 2 with the stopping criterion $\max_{x\in\mathbf{X}} |(h_{k+1}(x) - h_k(x))/h_k(x)| < 10^{-5}$ (254, 255 and 256 iterations were needed to compute the optimal policy displayed in Figures 2, 3 and 4, respectively). We see from these figures that the optimal policy is a *monotone switching curve*, namely, there exist two monotone (decreasing here) integer mappings $f_s : \{0, 1, \ldots, N\} \to \{0, 1, 2, \ldots\}$, $s \in \{0, 1\}$, such that $\pi^\star(x) = \mathbf{1}(f_s(r) \geq q)$ for all $x = (q, r, s) \in \mathbf{X}$ with $s = 0, 1$ (we must also have $f_0(0) \geq 1$ so that $\pi^\star(1, 0, 0) = 1$ as required). We conjecture that the optimal policy always exhibits such a structure but we have not able been to prove it.

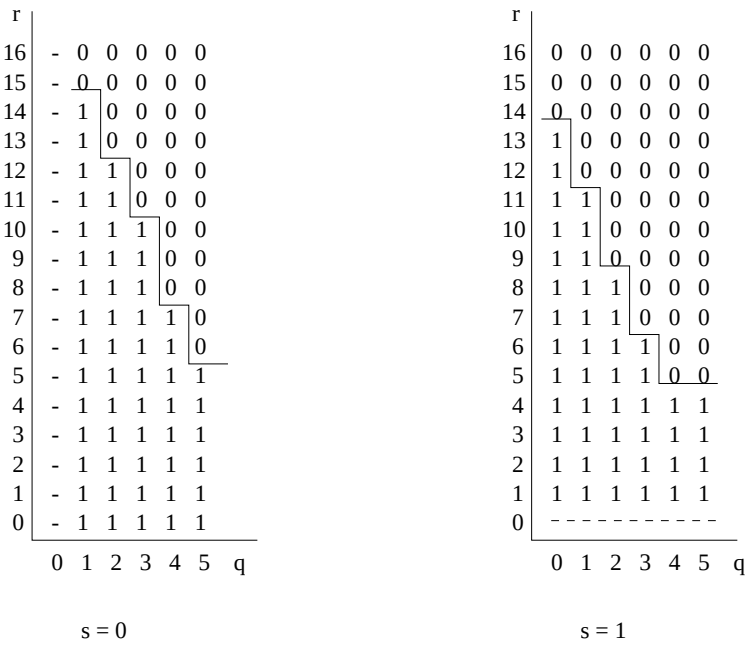

Figure 2: Optimal policy ($\gamma = 1.0$, Cost $= 0.20907$)

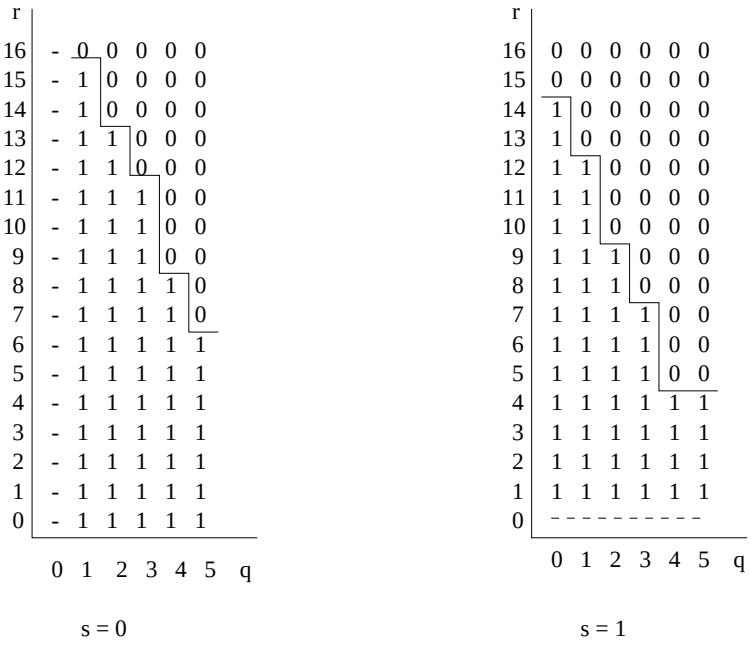

Figure 3: Optimal policy ($\gamma = 1.4$, Cost $= 0.25924$)

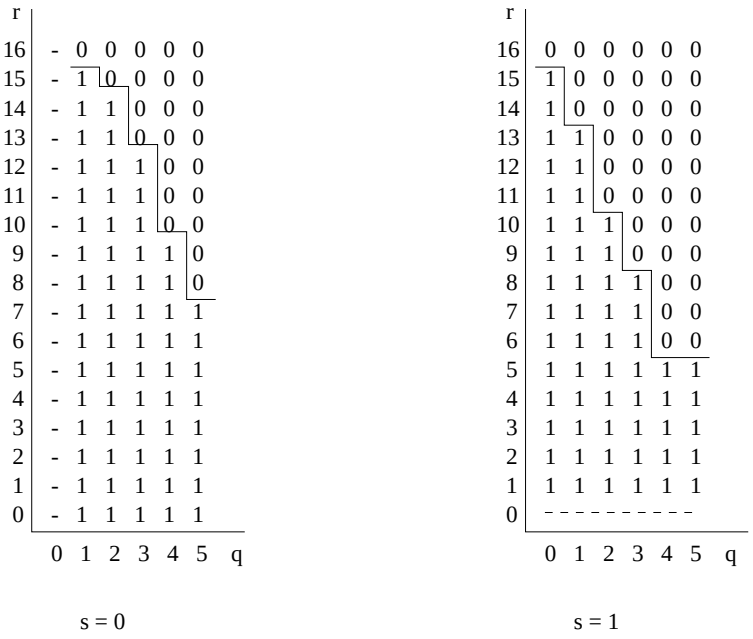

Figure 4: Optimal policy ($\gamma = 2.0$, Cost $= 0.32211$)

4 Static versus dynamic policies

In this section we compare static and dynamic policies. The results for the static case, with the same underlying queueing model and notation, are as follows (see [6]). The average cost is given by

$$C_s(\rho, \gamma, K) = \frac{(1-\rho)(\gamma + \mu\rho^{K+1})}{1-\rho^{K+1}}; \tag{9}$$

in particular, $C(\rho, \gamma, K) = (\gamma + \mu)/(K+1)$ when $\rho = 1$. The optimal number of robots is given by

Proposition 3 *For any $\gamma > 0$, $K \geq 2$, let $N_s(\gamma, K)$ be the optimal number of robots to use. Then,*

$$N_s(\gamma, K) = \arg\min_n C(n\lambda/\mu, \gamma, K) \tag{10}$$

with $n \in \{\lfloor \rho(\gamma, K)\mu/\lambda \rfloor, \lceil \rho(\gamma, K)\mu/\lambda \rceil\}$. Furthermore, $N_s(\gamma, K) \leq \lceil \mu/\lambda \rceil$ if $\gamma < \gamma(K)$, $N_s(\gamma, K) \in \{\lfloor \mu/\lambda \rfloor, \lceil \mu/\lambda \rceil\}$ if $\gamma = \gamma(K)$, and $N_s(\gamma, K) \geq \lfloor \mu/\lambda \rfloor$ if $\gamma > \gamma(K)$. ◇

The results of the comparisons are reported in Tables 1 and 2. Throughout the experiments $\mu = 1.0$. For different sets of parameters λ, K, γ, we first computed the optimal number of robots N_s (given by Proposition 3) and the average cost C_s (given in (9)) in the static case. Then, for each set of parameters λ, K, γ, we set the value of the number of available robots N to N_s and determined, via the relative value iteration algorithm given in Proposition 2 (with $\tau = 0.99999$ – the closer τ is to 1 the faster the algorithm converges), the optimal average cost C_d (given in (6)) as well as the minimun ($N_{\min}$) and the expected ($\overline{N}$) number of robots activated by the optimal *dynamic* policy. These results can be found in Table 1.

We stopped the numerical procedure when the relative error between two consecutive iterates was (uniformly) less than 10^{-5}. The number of iterations (N_{iter}) and the relative improvement ($100\% \times (C_s - C_d)/C_d$) are also reported in Table 1.

Last, we computed the overall optimal dynamic policy by removing the restriction on the number of available robots. The optimal average cost C_d as well as the minimum ($N_{\min}$), expected ($\overline{N}$) and maximum ($N_{\max}$) number robots used by the overall optimal dynamic policy are given in Table 2.

We observe that substantial gains may be achieved by dynamically controlling the activity of the robots. When the number of available robots is set to N_s (Table 1), the relative improvement with respect to the optimal static policy ranges from 4% to 103% for the considered model parameters; when the restriction on the number of available robots is removed, the improvement ranges from 6% to 3226%! The gain appears to be an increasing function of the queue size K and of the arrival rate λ.

			Static Approach		Dynamic Approach				
λ	K	γ	C_s	N_s	C_d	$N_{\min}$	$\bar{N}$	N_{iter}	Rel. Impr.
0.01	5	0.4	0.17541	73	0.16804	57	70.3	1634	4%
-	-	1.4	0.40000	100	0.38336	86	95.1	1911	4%
-	-	2.4	0.53834	114	0.51746	101	108.4	2051	4%
0.01	10	0.4	0.10207	86	0.09062	60	82.6	1794	13%
-	-	1.2	0.20000	100	0.17534	77	94.1	1939	14%
-	-	2.4	0.28347	110	0.24798	88	102.3	2039	14%
0.01	15	0.4	0.07177	91	0.05891	58	87.7	1860	22%
-	-	1.13	0.13313	100	0.10720	70	94.5	1953	24%
-	-	2.4	0.19192	107	0.15342	78	99.7	2024	25%
0.05	5	0.4	0.17578	15	0.15127	7	13.8	338	16%
-	-	1.4	0.40000	20	0.34733	12	17.7	391	15%
-	-	2.4	0.53841	23	0.46583	15	20.2	422	16%
0.05	10	0.4	0.10220	17	0.08308	5	16.2	369	23%
-	-	1.2	0.20000	20	0.14955	8	18.2	402	34%
-	-	2.4	0.28347	22	0.20541	10	19.4	423	38%
0.05	15	0.4	0.07184	18	0.05514	4	17.4	401	30%
-	-	1.13	0.13313	20	0.09117	6	18.7	426	46%
-	-	2.4	0.19372	21	0.13895	8	19.3	438	39%
0.1	5	0.4	0.17600	7	0.15239	1	6.5	167	15%
-	-	1.4	0.40000	10	0.32198	4	8.6	200	24%
-	-	2.4	0.54067	11	0.44989	5	9.3	211	20%
0.1	10	0.4	0.10403	9	0.06838	0	8.4	204	52%
-	-	1.2	0.20000	10	0.13854	2	9.0	218	44%
-	-	2.4	0.28347	11	0.18585	3	9.6	227	53%
0.1	15	0.4	0.07184	9	0.05326	0	8.7	312	35%
-	-	1.13	0.13313	10	0.08538	1	9.3	359	56%
-	-	2.4	0.19458	11	0.09606	1	9.7	376	103%

Table 1: Static vs. dynamic policies (with $\mu = 1.0$ and $\tau = 0.99999$)

			Static Approach		Dynamic Approach				
λ	K	γ	C_s	N_s	C_d	$N_{\min}$	$\bar{N}$	$N_{\max}$	Rel. Impr.
0.01	5	0.4	0.17541	73	0.16595	58	74.8	82	6%
-	-	1.4	0.40000	100	0.37886	88	99.9	115	6%
-	-	2.4	0.53834	114	0.51179	103	113.0	133	5%
0.01	10	0.4	0.10207	86	0.08124	62	89.3	105	27%
-	-	1.2	0.20000	100	0.15876	78	99.6	123	26%
-	-	2.4	0.28347	110	0.22777	89	107.1	137	24%
0.01	15	0.4	0.07177	91	0.04236	59	94.9	118	69%
-	-	1.13	0.13313	100	0.07812	71	99.8	131	70%
-	-	2.4	0.19192	107	0.11493	79	103.6	143	67%
0.05	5	0.4	0.17578	15	0.13770	7	15.9	20	28%
-	-	1.4	0.40000	20	0.31712	13	19.8	27	26%
-	-	2.4	0.53841	23	0.43292	16	21.9	32	24%
0.05	10	0.4	0.10220	17	0.04128	5	19.0	29	148%
-	-	1.2	0.20000	20	0.12020	8	19.9	33	66%
-	-	2.4	0.28347	22	0.20541	10	20.6	36	38%
0.05	15	0.4	0.07184	18	0.00969	2	19.8	35	641%
-	-	1.13	0.13313	20	0.01818	4	20.0	38	632%
-	-	2.4	0.19372	21	0.02782	6	20.1	41	596%
0.1	5	0.4	0.17600	7	0.11097	2	8.2	12	59%
-	-	1.4	0.40000	10	0.25924	4	9.8	16	54%
-	-	2.4	0.54067	11	0.35805	6	10.7	18	51%
0.1	10	0.4	0.10403	9	0.01937	0	9.7	18	437%
-	-	1.2	0.20000	10	0.03887	1	9.9	20	415%
-	-	2.4	0.28347	11	0.05894	2	10.1	22	381%
0.1	15	0.4	0.07184	9	0.00188	0	10.0	24	3721%
-	-	1.13	0.13313	10	0.00368	0	10.0	25	3518%
-	-	2.4	0.19458	11	0.00585	0	10.0	27	3226%

Table 2: Static vs. dynamic policies (with $\mu = 1.0$ and $\tau = 0.99999$)

5 Concluding remarks

A simple queueing model of search engines has been proposed and analyzed in order to find the optimal number of robots to deploy on the Web. The cost function is a weighted sum of the loss rate and the starvation probability.

We studied the dynamic setting in which the number of robots is allowed to change over time as a function of the workload in the queue. We showed that in most cases dynamic policies substantially out-perform static policies, and based on this fact, advocate the introduction of such policies in Web search engines.

Several interesting open issues remain, including models where the robots are not homogeneous and/or are allocated to different parts of the network. For instance, one may wish to determine

the optimal number of robots to be allocated to a given area. Also, more general input processes (e.g. a Markov modulated Poisson process) should be considered so as to reflect more accurately "traveling times" of robots in the network. Lastly, other cost functions could be investigated, for instance, cost functions including response times.

References

[1] Bertsekas, D. P., *Dynamic Programming. Deterministic and Stochastic Models*, Prentice-Hall, Inc., Englewood Cliffs, 1987.

[2] Coffman Jr., E. G., Liu, Z. and Weber, R. R., "Optimal robot scheduling for Web search engines", *J. Scheduling*, **1**, pp. 14-22, 1998.

[3] Kleinrock, L., *Queueing Systems, Vol. I*, Wiley & Sons, New York, 1975.

[4] Puterman, M. L., *Markov Decision Processes*, Wiley, New York, 1994.

[5] Ross, S. M., *Introduction to Stochastic Dynamic Programming*, Academic Press, New York, 1983.

[6] J. Talim, Z. Liu, Ph. Nain, and E. G. Coffman, Jr. "Optimizing the number of robots for Web search engines", *Telecommunication Systems*, vol. 17, pp. 245–266, 2001.

[7] Wolff, R. L., "Poisson Arrivals See Time Averages," *Oper. Res.*, vol. 30, pp. 223-231, 1982.

Appendix

A proof that the MDP $(x_n, n \geq 1)$ is communicating

Proof of Lemma 1. First, a word on notation. In the following $(\bullet, r, s)$ (respectively, $(q, r, \bullet)$) will designate any state $\hat{x} = (\hat{q}, \hat{r}, \hat{s}) \in \mathbf{X}$ such that $(\hat{r}, \hat{s}) = (r, s)$ (respectively, $(\hat{q}, \hat{r}) = (q, r)$). We will say that $x = (q, r, s)$ is at the same *level* as $x' = (q', r', s')$ if $r = r'$.

There are three cases in the proof depending on whether state $x' = (q', r', s')$ reached from $x = (q, r, s)$ is at the same level as x (case (1)), at a higher level (case (2)), or at a lower level (case (3)).

(1) $r = r'$. Assume first that $r > 0$. Then select any policy $\pi \in \mathcal{P}$ such that

$$\pi(\bullet, r, 0) = 0 \quad \text{and} \quad \pi(\bullet, r, 1) = 1.$$

Under that policy once the process enters level $r > 0$ it cannot leave that level, and moreover, all states of that level are recurrent.

Assume now that $r = 0$. If $q \geq q'$ then choose $\pi(i, 0, 0) = 0$ for $i = q, q+1, \ldots, q'+1$. If $q < q'$ then x' cannot be reached from x without jumping at level 1 since no arrival may occur at level 0. In this case, select a policy that goes from x to state $(0, 1, \bullet)$ ($\pi(i, 0, 0) = 0$ for $i = q, q+1, \ldots, 2$

and $\pi(1,0,0)=1$), then go to state $(q'-1,1,1)$ ($\pi(i,1,1)=1$ for $i=0,1,\ldots,q'-2$) and then go to state x' ($\pi(q'-1,1,1)=0$).

(2) $r<r'$.

Choose any policy $\pi\in\mathcal{P}$ such that

$$\pi(\bullet,t,0)=1 \quad \text{and} \quad \pi(\bullet,t,1)=1 \qquad \text{for } t=r,r+1\ldots,r'-1;$$
$$\pi(\bullet,r',0)=0 \quad \text{and} \quad \pi(\bullet,r',1)=1.$$

Under that policy the process will successively visit levels $r,r+1,\ldots,r'$ and will stay forever at that last level where it will visit all states infinitely often.

(3) $r>r'$.

If $r'>0$ choose any policy $\pi\in\mathcal{P}$ such that

$$\pi(\bullet,t,s)=0 \qquad \text{for } t=r,r-1\ldots,r'-1,\ s=0,1;$$
$$\pi(\bullet,r',0)=0 \quad \text{and} \quad \pi(\bullet,r',1)=1.$$

Under that policy the process will successively visit levels $r,r-1,\ldots,r'$ and will stay forever at that last level where it will visit all states infinitely often.

Assume now that $r'=0$. Under the policy $\pi(\bullet,t,\bullet)=0$ for $t=r,r-1,\ldots,2$ the process will go from x to level 1. Once level 1 is reached, it will then move to state $(K,1,1)$ ($\pi(i,1,1)=1$ for $i=1,2,\ldots,K-1$), then go to state $(K,0,0)$ ($\pi(K,1,1)=0$), and finally go to state x' ($\pi(i,0,0)=0$ for $i=K,K-1,\ldots,q'+1$). ∎