Strategies for Querying Information Agents

Prasad Chalasani, Somesh Jha, Onn Shehory and Katia Sycara

Robotics Institute, Carnegie Mellon University, Pittsburgh PA 15213-3890, USA

Abstract. In a simple cooperative MAS model where a collection of "querying agents" can send queries to a collection of "information agents", we formalize the problem of designing strategies so that the expected completion time of the queries is minimized, when every querying agent uses the same strategy. We devise a provably optimal strategy for the static case with no query arrivals, and show via simulations that the same strategy performs well when queries arrive with a certain probability. We also consider issues such as whether or not the expected completion time can be reduced by sending multiple copies of queries, or by aborting copies of answered queries.

1 Introduction

As the internet grows relentlessly, and multi-agent systems (MAS) proliferate, it becomes increasingly important to design algorithms for agents to use limited resources (such as time, memory, bandwidth, etc) efficiently. A badly designed scheme can easily lead to congestion and poor response times. A first step toward the design of good strategies is to consider simple models of agent interactions. Even though these models may be abstract, they can help identify important issues that will arise in realistic models. Morever, strategies devised under simple models can perform well in realistic settings.

With this viewpoint, in this paper we introduce a simple MAS model where there is a collection of *querying agents* (QA) and a collection of *information agents* (IA). The querying agents receive queries (from humans or other agents), which they must send to IAs to obtain an answer. We want to design a strategy for a QA to send queries to IAs so that the expected completion time of the queries is minimized. To see what kinds of issues arise in designing such strategies, suppose the loads of the IAs were observable, and that all IAs are capable of answering any query. Should the QA send its query to the least-loaded information agent? If this were the *only* QA in the system, this is obviously a good strategy. However in an MAS there are a large number of QAs, possibly much larger than the number of IAs. If *every* QA uses the above strategy, then it is no longer clear that this is the best one. For instance if the loads of the information agents are roughly the same, this would be a bad strategy, since the least-loaded information agent would tend to receive a disproportionately large number of queries, and all these queries would take longer to complete, on the average. However if one IA has a much lower load than every other IA, this could be a good strategy.

In this paper we examine the following type of question:

> If every QA were to use the *same* querying strategy, which strategy minimizes the expected completion time of the queries?

Since every QA uses the same strategy, we refer to it as a *symmetric* strategy. A natural symmetric strategy is the following *randomized* one: every query agent sends its query to an information agent chosen uniformly at random. This would be a good strategy if the information agents are more or less equally loaded, but what if they aren't? In this paper we examine this problem and design an optimal randomized symmetric strategy for this case of arbitrary loads, and assuming a static situation where each QA has one query and no new queries arrive. Our strategy has the appealing feature that IAs with higher load are less likely to receive queries. We also show by simulations that even in a dynamic model with new query arrivals, ours is a good strategy to follow.

Another important issue we examine is:

> If QAs send multiple copies of their queries to different IAs, does this reduce the expected completion time? Is there an optimal number of copies to send?

There is a tradeoff here between two opposing effects on the query completion time. The first is the *load effect*: multiple copies increase the load on the IAs, and every *individual* query copy takes longer on average to be answered. The second is the *multiplicity effect:* since each query has multiple copies at different IAs, it has a greater chance of being answered sooner. It seems intuitive that as more copies are sent, the benefit of multiplicity will be outweighed by the load effect. In this paper we show simulations that show the optimal number of query copies to send, for specific situations.

1.1 Related work

Problems related to ours have been studied extensively in the area of *stochastic scheduling* of parallel systems (for a good introduction, see [2, 5, 6] and the references therein). In all such work, the goal has been to design a *centralized* scheduling algorithm so that job completion times are reduced. By contrast in our MAS model, we emphatically want to design *decentralized* algorithms that different querying agents can use, with as little communication as possible between each other. Decentralized algorithms are easy to implement, more robust in the face of failures, and scale up better than centralized ones. Such algorithms are therefore likely to play an important role in a MAS context. Despite this important difference, some of the techniques in stochastic scheduling research are useful for our purposes. For instance, we have used the concepts of *majorization* and *Schur convexity* [1, 3, 9] to design our optimal randomized algorithm in Section 4.

Several researchers in MAS have approached the problem of designing decentralized strategies from an *economics* viewpoint. For instance, Huberman and Lukose [11] have observed that since most people who access the internet are not charged in proportion to their use, this has lead to the well-known *tragedy of the commons* [8], which is a special kind of *social dilemma*: each individual tends to greedily consume bandwith, leading to a degradation of performance for everyone. This conflict between an individual's myopic strategy and global performance is similar to the one discussed above in the introduction: when every QA sends a query to the least-loaded IA, everyone's performance suffers. Researchers taking the economic viewpoint have proposed that *pricing* internet access can lead to a resolution of this dilemma [12, 14, 17, 18]. Some researchers [10, 15, 19, 20] are pursuing the design of decentralized strategies using models based on *market equilibrium* [7]. In this paper we are formulating the decentralized strategy-design problem purely from a performance viewpoint: if each agent uses a strategy that leads to degraded performance for every agent, then that strategy is perhaps not an optimal one.

Querying strategies for *individual agents* have been considered by, among others, Chalasani et. al. [4] Etzioni et. al. [16], and Lukose and Huberman [13]. These authors have not considered the effect of several agents using the same strategies.

1.2 Organization of the paper

In Section 2 we introduce the basic model assumed throughout the paper. In Section 3 we consider the case where the IAs initially have zero load. For this case we show a lower bound on the expected completion time of *any* strategy. We also design a randomized algorithm that comes close to the lower bound. In Section 4 we consider the case where IAs have arbitrary initial loads and design an optimal symmetric randomized querying strategy. We also show via simulations (subsection 4.1) that this strategy performs better than two other natural ones. Section 5 examines the effect of sending multiple query-copies to different IAs. We show analytical results for some cases, and simulations for others. Section 6 concludes with a discussion of future work.

2 The model

We assume there are m **querying agents** (QA) $A_1, A_2, \ldots, A_m$, and n **information agents** (IA) $I_1, I_2, \ldots, I_n$. Initially, each IA I_i has a **load** ℓ_i, that is, it has ℓ_i queries pending, and, without loss of generality,

$$\ell_1 \leq \ell_2 \leq \ldots \leq \ell_n.$$

Time is measured in **cycles**, and the initial cycle represents time 0. In general, the QAs receive queries that they need to send to IAs for an answer. Every IA is capable of answering every query. In the **static** version of the model, each QA has just one query at time 0. In the **dynamic** version, queries arrive at each QA in each cycle with a certain **arrival probability** α. Each QA can send up to k **copies** (or instances) of the query to different IAs. The query is said to be **completed** as soon as any copy of the query is answered by an IA. Queries never fail, i.e., when an IA chooses to answer a certain query, it successfully does so. Each query takes exactly one cycle to answer. Each IA uses the following **randomized scheduling** policy: Among the queries that are pending, it picks one uniformly at random and answers it, and deletes it from its pending list. Note that under this policy, if the number of pending queries at an IA is large, then *every* query at this IA experiences a longer *expected* completion time. When a QA's query has been answered, the QA may choose to **abort** all (unanswered) copies of its query. We ignore all communication costs and assume that queries and answers are sent instantaneously.

Our goal is to design a good **symmetric** strategy for the QAs to send queries (with possibly multiple copies) to the IAs. By a symmetric strategy we mean that every QA uses exactly the same strategy. In addition to being easy to analyze, such strategies are also easy to implement in a cooperative multi-agent system (MAS) setting. In this paper we will only consider the design of strategies for the static model, and experimentally study the behavior of the dynamic model when each QA uses this static strategy in each cycle.

Consider then the static model, where each QA has just one query at time 0, that it wants to obtain an answer for. For brevity we refer to QA A_i's query simply as "query i". For any (possibly randomized) symmetric strategy, we define the following random variables. We let X_{ij} be the random variable defined as

$$X_{ij} = \begin{cases} 1 & \text{if a copy of query } i \text{ is sent to IA } I_j, \\ 0 & \text{otherwise.} \end{cases}$$

If A_i sends a total of k copies of its query, then clearly

$$\sum_{j=1}^{n} X_{ij} = k.$$

Y_{ij} is the time at which a copy of query i is answered by IA I_j, if it received such a copy, and is ∞ otherwise. The **completion time** of query i is the random variable Z_i defined as

$$Z_i = \min\{X_{i1}Y_{i1}, X_{i2}Y_{i2}, \ldots, X_{in}Y_{in}\}.$$

In case only $k = 1$ copy of query is i is sent, to I_j, then of course $Z_i = Y_{ij}$.

3 Single query-copy, unloaded case

We consider first the simplest case of the static model where the initial loads ℓ_i of the IAs are all 0, and each QA sends exactly *one* copy of its query to some information agent (so $k = 1$). What symmetric strategy should the QAs use in order to minimize the expected completion time of their query? We first show a lower bound on the expected completion time, for *any* strategy (symmetric or not).

Lemma 1. *For the static model where each QA sends exactly one copy of its query to an IA, regardless of the strategy used, there is some query whose expected completion time is at least*

$$\left(\lfloor \frac{m}{n} \rfloor + 1\right)\left(1 - \frac{n}{2m}\lfloor \frac{m}{n} \rfloor\right).$$

If m is a multiple of n, this simplifies to

$$\frac{1}{2}\left(1 + \frac{m}{n}\right).$$

Proof: Following the notation introduced above, we write Z_i for the completion time of the query sent by A_i. Consider the *sum* of the completion times of the m queries $Z = Z_1 + Z_2 + \ldots + Z_m$. This sum depends on the actual allocation of the m queries among the n IAs. What is the smallest possible value of this sum? Clearly Z is minimized if, for as many cycles as possible, *every* IA is busy answering some query. If m is an integer multiple of n, this is easily achieved by allocating exactly m/n of the queries to each IA. In general, the minimum Z is achieved by allocating $\lfloor m/n \rfloor$ queries to each IA, and allocating each of the remaining $m - n\lfloor m/n \rfloor$ queries to distinct IAs. With this allocation, in cycles 1 to $\lfloor m/n \rfloor$, n different queries are answered in each cycle, and in last cycle number $\lfloor m/n \rfloor + 1$, the remaining $m - n\lfloor m/n \rfloor$ queries are answered. Thus for any querying strategy

$$\begin{aligned}
\sum_{i=1}^{m} Z_i &\geq \sum_{i=1}^{\lfloor \frac{m}{n} \rfloor} (ni) + \left(m - n\lfloor \frac{m}{n} \rfloor\right) \left(\lfloor \frac{m}{n} \rfloor + 1\right) \\
&= \frac{n}{2}\lfloor \frac{m}{n} \rfloor \left(\lfloor \frac{m}{n} \rfloor + 1\right) + \left(m - n\lfloor \frac{m}{n} \rfloor\right) \left(\lfloor \frac{m}{n} \rfloor + 1\right) \\
&= \left(\lfloor \frac{m}{n} \rfloor + 1\right) \left(m - \frac{n}{2}\lfloor \frac{m}{n} \rfloor\right).
\end{aligned}$$

By linearity of expectations, $\sum_{i=1}^{m} \mathbf{E}Z_i$ is also lower bounded by the last expression above. By the pigeonhole principle, this implies there is some i such that $\mathbf{E}Z_i$ is at least $1/m$ times that expression, which is the desired lower bound. ■

A simple and natural strategy that comes to mind is the following randomized one: Each QA sends its query to an IA chosen *uniformly at random.* It is clear that every query has the same expected completion time, and we show that this comes very close to the above lower bound.

Lemma 2. *If each QA sends its query to a uniformly randomly chosen IA, the expected completion time for each query is*

$$1 + (m-1)/(2n).$$

Proof: Since every query has the same expected completion time, without loss of generality we can focus on A_1's query. We let V be the random variable denoting the index of the information agent to which A_1 sends its query. The completion time Z_1 of A_1's query at I_V depends on the number of *other* queries I_V receives, which we denote by N_V. In particular, the query could be answered at times $1, 2, \ldots, N_V + 1$, each being equally likely. Therefore the conditional expectation of Z_1 given V is

$$\mathbf{E}(Z_1|V) = \frac{1}{N_V + 1} \sum_{i=1}^{N_V+1} i = (N_V + 2)/2 = 1 + N_V/2,$$

Note that, regardless of the value of the index V, $\mathbf{E}N_V = (m-1)/n$ since each of the $m - 1$ other QAs independently sends a query to I_V with probability $1/n$. So the expectation of Z_1 is

$$\begin{aligned}
\mathbf{E}(Z_1) &= \mathbf{E}[\,\mathbf{E}(Z_1|V)\,] \\
&= 1 + \frac{1}{2}\mathbf{E}(N_V) \\
&= 1 + (m-1)/(2n).
\end{aligned}$$

■

Example. Suppose there are $m = 17$ QAs and $n = 4$ IAs. With the above uniform-random strategy, the expected completion time of any query is, from Lemma 2,

$$1 + (m-1)/2n = 1 + 16/8 = 3.$$

The lower bound on the expected completion time is, from Lemma 1,

$$(1+4)(1 - 4 \times 4/34) = 2.64,$$

so our strategy comes close to the theoretical lower bound. In fact we conjecture that there is no *symmetric* strategy that can do better than the above uniform-random strategy.

4 Single query-copy, pre-loaded case

We continue to assume each QA sends a single copy of its query to some IA, but drop the assumption that the current loads ℓ_i of the IAs are 0. For this case we design an optimal randomized symmetric querying strategy that has an intuitive feature: IAs with a larger load are less likely to receive a query. Specifically, we want to design the following type of strategy for the QAs: The QA sends its query to an I_i with probability p_i. Our goal is to specify the p_i values in such a way that the expected completion time of any query is minimized.

We first derive an expression for the expected completion time of a query, in terms of the probabilities p_i. Since every query will have the same expected completion time, it suffices to consider the completion time of QA A_1's query. However, unlike the unloaded situation, the expected completion of this query *does* depend on which IA it is sent to. As before we let the random variable V denote the IA index to which A_1's query is sent, so that V takes values in $\{1, 2, \ldots, n\}$. We also let N_V denote the number of *other* queries, i.e., from $A_2, \ldots, A_m$, that land at I_V. Given that A_1's query lands at I_V, the completion time of A_1's query is one of $1, 2, \ldots, \ell_V + N_V + 1$, each being equally likely, and the expected completion time is

$$\mathbf{E}(Z_1|V) = \frac{1}{1+\ell_V+N_V} \sum_{j=1}^{\ell_V+N_V+1} = 1 + (\ell_V + N_V)/2,$$

Therefore the expected completion time of A_1's query is

$$\begin{aligned}
\mathbf{E}(Z_1) &= \mathbf{E}[\mathbf{E}(Z_1|V)] && (1)\\
&= 1 + \frac{1}{2}\mathbf{E}\left(\ell_V + N_V\right)\\
&= 1 + \frac{1}{2}\left(\sum_{i=1}^{n} p_i\ell_i + \mathbf{E}(N_V)\right)\\
&= 1 + \frac{1}{2}\left(\sum_{i=1}^{n} p_i\ell_i + \mathbf{E}[\mathbf{E}(N_V|V)]\right)\\
&= 1 + \frac{1}{2}\left(\sum_{i=1}^{n} p_i\ell_i + \mathbf{E}[(m-1)p_V]\right)\\
&= 1 + \frac{1}{2}\left(\sum_{i=1}^{n} p_i\ell_i + \sum_{i=1}^{n} p_i^2(m-1)\right)\\
&= 1 + \frac{1}{2}\left(\sum_{i=1}^{n} p_i[\ell_i + (m-1)p_i]\right). && (2)
\end{aligned}$$

From this expression it follows that:

Lemma 3. *The optimal choice of probabilities p_i satisfies*

$$p_1 \geq p_2 \geq \ldots \geq p_n, \qquad (3)$$

and in particular there is some k such that $p_i > 0$ for all $i \leq k$ and $p_i = 0$ for all $i > k$.

Proof: The proof is by contradiction. Suppose $p_1, \ldots, p_n$ is an optimal assignment of probabilities. If $p_i < p_{i+1}$ for some $i < n$, we can interchange p_i and p_{i+1} in the expression (2) and the expectation would decrease (since $\ell_i \leq \ell_{i+1}$), which contradicts our assumption that the probabilities were optimal. ■

How do we find the number k of positive probabilities in the optimal assignment? This turns out to be a non-trivial problem. We have the following result, whose proof appears in the appendix. First we introduce the symbols

$$\begin{aligned}
L(k) &= \sum_{i=1}^{k} \ell_k,\\
A(k) &= \left(L(k)/2 + m - 1\right)/k. && (4)
\end{aligned}$$

Theorem 4. *The number of positive probabilities in the optimal assignment is the smallest value of $k < n$ for which $A(k) \leq \ell_{k+1}/2$ if such a k exists, and equals n otherwise. For this k, the optimal probabilities p_i are such that for all $i \leq k$,*

$$\ell_i/2 + (m-1)p_i = A(k),$$

so that

$$p_i = \frac{1}{(m-1)}\left(A(k) - \ell_i/2\right). \qquad (5)$$

Note that the optimal probabilities have a nice intuitive feature: *IAs with larger loads have a lower probability of receiving a query.*

Example. We illustrate the computations of this section with a simple example. Suppose there are $m = 20$ QAs, and $n = 9$ IAs, with initial loads ℓ_i as follows:

$$\{\ell_1, \ell_2, \ldots, \ell_9\} = \{1, 3, 4, 7, 11, 12, 16, 24, 32\}.$$

The values of $A(k)$, $k = 1, 2, \ldots, 9$ are, from (4),

$$\{19.5, 10.5, 7.67, 6.63, 6.4, 6.33, 6.57, 7.25, 8.22\},$$

and we see that the smallest k for which $A_k \leq \ell_{k+1}/2$ is $k = 6$. Therefore in the optimal solution, probabilities p_1 through p_6 are positive, and $\ell_i/2 + (m-1)p_i$ has has the same value for $i = 1, 2, \ldots, 6$. The corresponding optimal probabilities are, from (5),

$$\{p_1, p_2, \ldots, p_6\} = \{0.307, 0.254, 0.228, 0.149, 0.0438, 0.0175\},$$

and the expected completion time $\mathbf{E}(Z_1)$ is 5.096 cycles. By contrast, if we had used the uniform-random strategy of the previous section (all $p_i = 1/n$), the expected completion time would be 8.167.

4.1 Simulation in the dynamic model

So far we have worked in the static model, i.e., each QA has just one query that needs to be answered. Now we consider the dynamic model, where in each cycle, at each QA, a new query arrives with probability α. To be realistic we also assume a bound M on the **buffer** at each QA, that is, each QA may have no more than M unanswered queries at any time.

For this model we consider what happens if each QA follows a specific static strategy in each cycle. In particular we consider the following three strategies:

– OPT: Each QA sends its query according the optimal static strategy of section 4, with the loads ℓ_i equal to the current loads of the IAs. Note that we
– MIN: Each QA sends its query to the *least-loaded* IA.
– UNIF: Each QA sends its query to an IA chosen uniformly at random.

Note that the first two strategies assume that the IA loads can be observed by the QAs, whereas the UNIF strategy does not require this capability. Figure 1 is a plot showing how these strategies compare with each other. We find that for the most part, our strategy OPT dominates the others.

5 Multiple query copies, unloaded case

Returning to the static model, let us examine strategies where each QA sends k copies of its query to a set of k distinct IAs, where the k-subset is chosen uniformly at random over all possible k-subsets. Note that since we are only considering symmetric strategies, the parameter k is the same for all QAs. The analytic computation of the expected query completion time in this case is somewhat complicated, and we will only consider

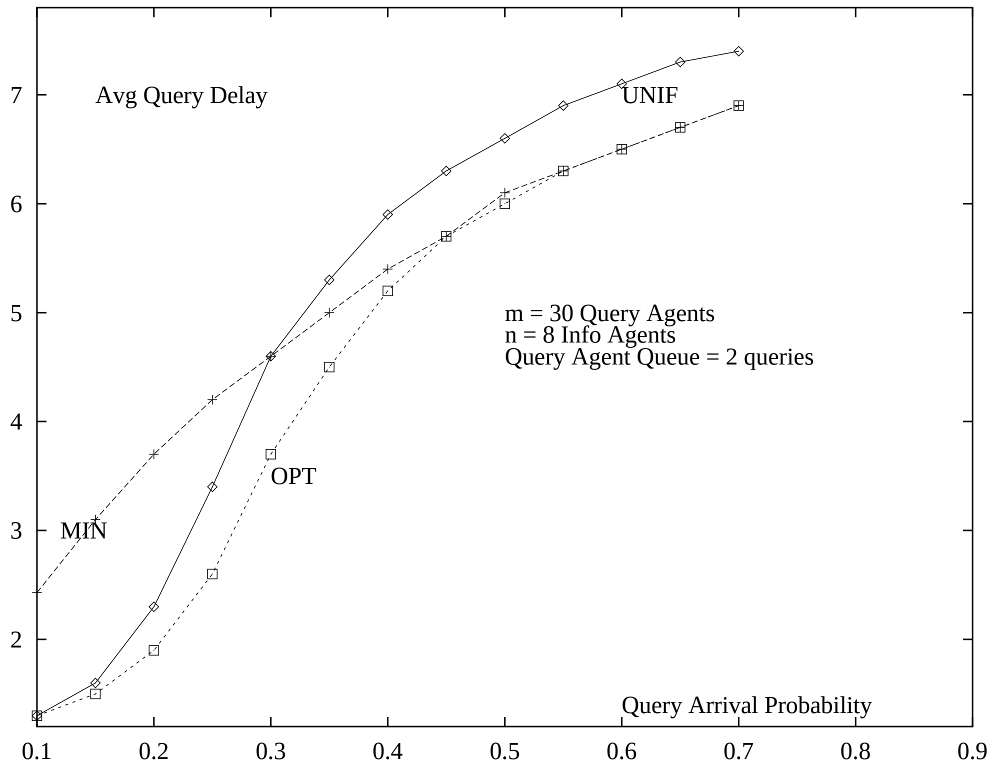

Fig. 1. *Expected completion time of a query in the dynamic model under three different strategies, as a function of the query arrival probability α. The model has $m = 30$ Query Agents, $n = 8$ Information Agents, and the buffer at each QA is limited to 2 unanswered queries. The average completion time is computed over all queries that have completed by 5000 runs.*

the special case $k = n$, i.e., an instance of *each* query is sent to *every* IA. For other k, we will only present simulation results.

For $k = n$, we consider two cases. First we will consider the case where *copies of answered queries are not aborted.* We show the following:

Theorem 5. *In the static model, if each of the m QAs sends a copy of its query to each of the n IAs, and copies of answered queries are not aborted, the expected completion time of any query approaches (for large m and n)*

$$\frac{1 - e^{-n}}{1 - e^{-n/m}}.$$

Proof: As before we focus on query 1 (i.e. QA A_1's query) and consider its completion time. We let Y_i denote the time at which the copy of query 1 sent to IA I_i is answered. Then clearly the completion time of query 1 is

$$Z_1 = \min\{Y_1, Y_2, \ldots, Y_n\}.$$

Therefore the expected completion time of query 1 is

$$\begin{aligned}
\mathbf{E}(Z_1) &= \sum_{i=1}^{m} P(Z_1 \geq i) \\
&= \sum_{i=1}^{m} P(Y_1 \geq i,\ Y_2 \geq i, \ldots, Y_n \geq i),
\end{aligned}$$

and since the random variables Y_i are independent, this is

$$\begin{aligned}
&= \sum_{i=1}^{m} P(Y_1 \geq i)^n \\
&= \sum_{i=1}^{m} \left(1 - \frac{i-1}{m}\right) && (6) \\
&= \sum_{i=1}^{m} \left[\left(1 - \frac{i-1}{m}\right)^{m/(i-1)}\right]^{(i-1)n/m} \\
&\simeq \sum_{i=1}^{m} e^{-(i-1)n/m} \quad \text{(for large } m, n\text{)} \\
&= \frac{1 - e^{-n}}{1 - e^{-n/m}}. && (7)
\end{aligned}$$

■

Now let us consider the case where *all copies of answered queries are aborted.* This case is more complicated to analyze because we have to carefully keep track of how many *distinct* queries are answered in each cycle. We assume a *synchronous* mode of operation, i.e., in each cycle, *first* each IA answers a randomly chosen query from its pending list, and *then* all copies of answered queries are removed from the lists. We then have the following result, whose proof is in the appendix.

Theorem 6. *In the static model, if each of the m QAs sends a copy of its query to each of the n IAs, and copies of answered queries are aborted, the expected completion time E_m of any query is given by the recursion: $E_j = 1$ for $j = 1$, and for $j > 1$,*

$$E_j = 1 - (1 - 1/k)^n + \sum_{i=1}^{\min\{j-1,n\}} \sum_{r=0}^{i} (-1)^r \binom{j-1}{i} \binom{i}{r} \left(\frac{i-r}{j}\right)^n (E_{j-i} + 1).$$

Simulations. In Fig. 2 we show how the average query delay changes as the number of query-copies k is increased. Interestingly, for many cases it is found that the query first decreases as k is increased, and then increases, indicating that there is a certain optimum number k of query-copies. As noted in the introduction, there are two opposing effects on the expected query completion time: the multiplicity effect, and the load effect. Clearly the initial decrease in expected completion time can be explained by the fact that the multiplicity effect dominates, and the subsequent increase occurs because the load effect starts to dominate.

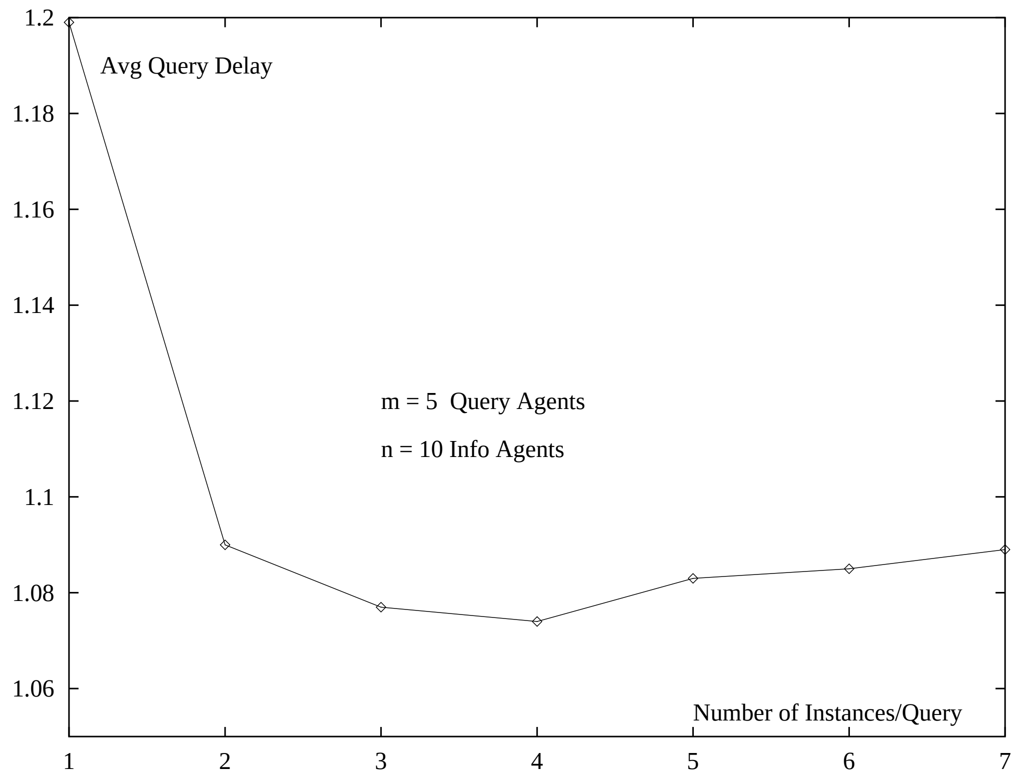

Fig. 2. *Expected completion time of a query, as a function of the number k of copies of the query that are sent. The plot is based on a simulation of the static model with $m = 5$ Query Agents, $n = 10$ Information Agents, and the average completion is computed over 5000 runs.*

6 Conclusion

In this paper we introduced a simple cooperative MAS model where there is a collection of information agents (IA) and a collection of querying agents (QA) that can send queries to the IAs. We designed a provably optimal randomized symmetric strategy for the static case where each QA has one query and each IA has an arbitrary initial load. We considered the issue of when it helps to send multiple query copies to different IAs. This paper only represents an initial step in a potentially fruitful and important research area, namely the design of decentralized algorithms for multi-agent systems. In the future we plan to study the use of economics-based approaches and also explore connections with the area of stochastic scheduling.

References

1. A.W.Marshall and I.Olkin. *Inequalities: theory of majorization and its applications*. Academic Press, 1979.

2. F. Baccelli, Z. Liu, and D. Towsley. Extremal scheduling of parallel processing systems with and without real-time constraints. *Journal of the ACM*, 40:1209–1237, 1993.
3. E. Beckenbach. *Inequalities*. Springer-Verlag, 1965.
4. P. Chalasani, S. Jha, O. Shehory, and K. Sycara. Query restart strategies for web agents. In *Autonomous Agents*, 1997. Submitted.
5. C. Chang and D. Yao. Rearrangement, majorization and stochastic scheduling. Technical Report IBM RC 16250 (#72136), IBM, Nov 1990.
6. E. Coffman and Z. Liu. On the optimal stochastic scheduling of out-forests. *Operations Research*, 40:S67–S75, 1992.
7. D. Duffie. *Security Markets: Stochastic Models*. Academic Press, 1988.
8. G. Hardin. The tragey of the commons. *Science*, 162:1243–1248, 1968.
9. G. Hardy, J. Littlewood, and G. Polya. *Inequalities*. Cambridge University Press, 1934.
10. B. Huberman, R. Lukose, and T. Hogg. An economics approach to hard computational problems. *Science*, 275:51–54, 1997.
11. B. A. Huberman and R. M. Lukose. Social dilemmas and internet congestion. *Science*, 277:535–537, July 25 1997.
12. J.K.MacKie-Mason and H. R. Varian. Some economics of the internet. In *Proc. 10th Michigan Public Utility Conf.*, 1993.
13. R. Lukose and B. Huberman. A methodology for managing risk in electronic transactions over the internet. In *3rd Int. conf. computational economics*, 1997.
14. J. MacKie-Mason and H. Varian. Pricing the internet. In B. Kahin and J. Keller, editors, *Public access to the internet*. MIT Press, 1995.
15. T. Mullen and M. Wellman. A simple computational market for network information services. In *Proc. first Int. Conf. on Multiagent Systems (ICMAS)*, 1995.
16. O.Etzioni, S. Hanks, T. Jiang, R. Kark, O. Madani, and O. Waarts. Efficient information gathering on the internet. In *Proc. Foundations of Comp. Sc.*, 1996.
17. D. Stahl, A. Gupta, and A. Whinston. Pricing of services in the internet. Technical report, University of Texas at Austin, 1995.
18. H. Varian. Economic mechanism design for computerized agents. In *USENIX Workshop on Electronic Conference*, New York, July 1995.
19. M. Wellman. A market-oriented programming environment and its application to distributed multicommodity flow problems. *J. Artificial Intelligence*, 1:1–23, 1993.
20. M. Wellman. The economic approach to artificial intelligence. *ACM Computing Surveys Symp. on Artif. Intell.*, 27(3), 1995.

A Proof of Theorem 4

Let us first introduce the variables

$$e_i = \ell_i/2 + (m-1)p_i, \quad i = 1, 2, \dots, n,$$

and rewrite the expression (2) for $\mathbf{E}(Z_1)$ as

$$\begin{aligned}\mathbf{E}(Z_1) &= \frac{1}{m-1}\sum_{i=1}^{n}(e_i + \ell_i/2)(e_i - \ell_i/2) \\ &= \frac{1}{m-1}\sum_{i=1}^{n}(e_i^2 - \ell_i^2/4). \end{aligned} \tag{8}$$

Since the p_i are probabilities that add up to 1, we must have

$$\ell_i/2 \leq e_i \leq \ell_i/2 + (m-1), \quad i = 1, 2, \ldots, n \tag{9}$$

$$\sum_{i=1}^{n} e_i = L(n)/2 + m - 1 \tag{10}$$

Thus our original problem of choosing the optimal probabilities can be recast as one choosing the $\{e_i\}$ so that the expectation (8) is minimized subject to the constraints (9) and (10). Suppose there are two distinct indices i, j such that (a) p_i and $p_j > 0$ are strictly positive in the optimal solution, and (b) $e_i < e_j$. Clearly this means e_i and e_j lie strictly within the range defined by (9). Therefore there is some sufficiently small $\epsilon > 0$ such that if we increase e_i by ϵ and decrease e_j by ϵ, we still satisfy the constraints (9) and (10). However the value of $e_i^2 + e_j^2$ is smaller, since, in general for any positive x, y,

> When $x + y$ is fixed, $x^2 + y^2$ is smaller when x and y are "closer" to each other, i.e., when $|x - y|$ is smaller.

(This type of argument is a special case of a *majorization* argument for *Schur-convex* functions (see, for example [1, 9]).) This means the expectation (8) is smaller, a contradiction. Therefore we conclude that for all positive p_i in the optimal solution, the value of e_i must be the same. A similar argument shows that if $i < j$ and $p_i = 0$ in the optimal solution, then $p_j = 0$ as well. Therefore if k is the number of positive probabilities in the optimal solution, for each $i \leq k$ the value of e_i is the same, and for $i > k$, $e_i = \ell_i/2$. Since $\sum_{i=1}^{k} e_i = L(k)/2 + m - 1$, it follows that for $i \leq k$, $e_i = A(k)$.

Suppose k is the number of positive probabilities in the optimal solution. Clearly if $A(k) > \ell_{k+1}/2$, this means $e_k > e_{k+1} = \ell_{k+1}/2$. As before this means we can decrease e_k by a small $\epsilon > 0$ and increase e_{k+1} by ϵ and reduce the sum $e_k^2 + e_{k+1}^2$ while still satisfying the constraints (9) and (10). This is a contradiction, so $A_k \leq \ell_{k+1}/2$. Now we claim that if

$$A_j \leq \ell_{j+1}/2$$

for some j, then this continues to hold for all larger j. To see this, note that

$$\begin{aligned}
A(j+1) &= \frac{L(j+1)}{2(j+1)} + \frac{m-1}{j+1} \\
&= \frac{1}{j+1}\left(j\left(L(j)/(2j) + (m-1)/j\right) + \ell_{j+1}/2\right) \\
&= \frac{1}{j+1}\left(jA(j) + \ell_{j+1}/2\right) \\
&\leq \frac{1}{j+1}\left(j\ell_{j+1}/2 + \ell_{j+1}/2\right) \\
&= \ell_{j+1}/2 \\
&\leq \ell_{j+2}/2.
\end{aligned}$$

Therefore the number k of positive probabilities in the optimal solution is the *smallest* $k < n$ such that $A(k) \leq \ell_{k+1}/2$ is such a k exists, or equals n otherwise. ■

B Proof of Theorem 6

Again we fix our attention on the completion time of query 1. Let E_j denote the expected completion time of query 1 when there are j distinct queries remaining. Initially, we of course have $j = m$. Clearly if $j = 1$ we have $E_1 = 1$. For $j > 1$, we have the following mutually exclusive and exhaustive events: Event A_0: Query 1 is answered in the current cycle, in which case the expected time is 1,a and the probability of event A is one minus the probability that none of the IAs answer query 1, i.e.,

$$\mathbf{P}(A) = 1 - (1 - 1/k)^n .$$

The remaining events are A_i for $i = 1, 2, \min\{j-1, n\}$, where A_i is the event that exactly i distinct queries are answered in the current cycle, all different from query 1. If event E_i occurs, all copies of the i answered queries will be removed, so we are left with $j - i$ distinct queries. Thus the expectation given that event A_i occurs is $(1 + E_{j-i})$. So we can write, for $j > 1$,

$$E_j = \mathbf{P}(A_0) + \sum_{i=1}^{\min\{j-1,n\}} \mathbf{P}(A_i)(1 + E_{j-i}).$$

We only need to show how to compute the probabilities $\mathbf{P}(A_i)$. We can write this as

$$\begin{aligned}\mathbf{P}(A_i) = &\text{ (Number of ways of choosing } i \text{ special queries out of } j-1)\\ &\times \mathbf{P}(\text{each IA picks only among the } i \text{ special queries})\\ &\times \mathbf{P}(\text{each of the } i \text{ special queries is picked by some IA}),\end{aligned} \quad (11)$$

which is

$$\mathbf{P}(A_i) = \binom{j-1}{i}\left(\frac{i}{j}\right)^n P_i,$$

where P_i is the last probability in (11). We compute this probability by considering the complementary event: the event that at least one of the i special queries is *not* picked by any IA. For $r \le i$ the probability that a particular r-subset of the i special queries are not picked by any IA is $(1 - r/i)^n$. By inclusion-exclusion, we then have

$$P_i = \sum_{r=0}^{i} (-1)^r \binom{i}{r} (1 - r/i)^n .$$

Thus finally the recursive formula for our expectation E_j is: if $j = 1$ then $E_j = 1$, and if $j > 1$,

$$E_j = 1 - (1 - 1/k)^n + \sum_{i=1}^{\min\{j-1,n\}} \binom{j-1}{i}\left(\frac{i}{j}\right)^n P_i(1 + E_{j-i}),$$

which simplifies to

$$E_j = 1 - (1 - 1/k)^n + \sum_{i=1}^{\min\{j-1,n\}} \sum_{r=0}^{i} (-1)^r \binom{j-1}{i}\binom{i}{r}\left(\frac{i-r}{j}\right)^n (E_{j-i} + 1).$$

■

This article was processed using the LaTeX macro package with LLNCS style