A Decision Theoretic Approach to Shopbot Design

Kartik Hosanagar[+], Ramayya Krishnan[#]

Abstract:

Shopbots are software agents that automatically query a large number of vendors for price and availability of products. They are the primary tools for comparison-shopping on the Internet. Despite their apparent usefulness, shopbots impose considerable costs on their users because of the manner in which they make their operational decisions. Specifically, search tends to be slow resulting in high wait times for users. In addition, user cognitive costs are high because shopbots present all possible results, several of which are irrelevant or are dominated. This paper models the shopbot decision problem of how long to wait for store responses and what offers to display as an optimization problem. The problem is a large stochastic integer programming problem. Thus, a key challenge is to address the uncertainty and discrete choices in real-time decision making. Given the complexity of the problem, we focus on the design of real-time algorithms that combine simulations and optimization techniques to compute near-optimal results. The algorithms are tested using data collected from 30 stores for a set of 60 books from the New York Times bestsellers list. Our research demonstrates how a shopbot can leverage information about price and response time distributions to develop intelligent algorithms and optimize its performance. We find that the dollar value of the gains to a shopbot user from such an algorithm can be close to 10% of the price paid by the user.

[+] Operations and Information Management Department, Wharton School of Business, University of Pennsylvania

[#] Heinz School of Public Policy and Management, Carnegie Mellon University

1. Introduction

Comparison shopping engines or shopbots are software agents that automatically query a large number of online vendors for price and availability of products. For example, a search for the book "My Life" by Bill Clinton resulted in 54 offers at Shopping.com and 14 offers at mySimon.com. Shopbots typically return price, store rating, availability information, and other product attributes in their display set (see Figure 1). A customer can follow one of the displayed hyperlinks to purchase the product directly at a store website. Shopbots have been touted to be the "enablers of frictionless commerce" [] because of their role in reducing consumer search costs. In contrast to traditional sequential search for price and product information, shopbots enable consumers to gather required information at once.

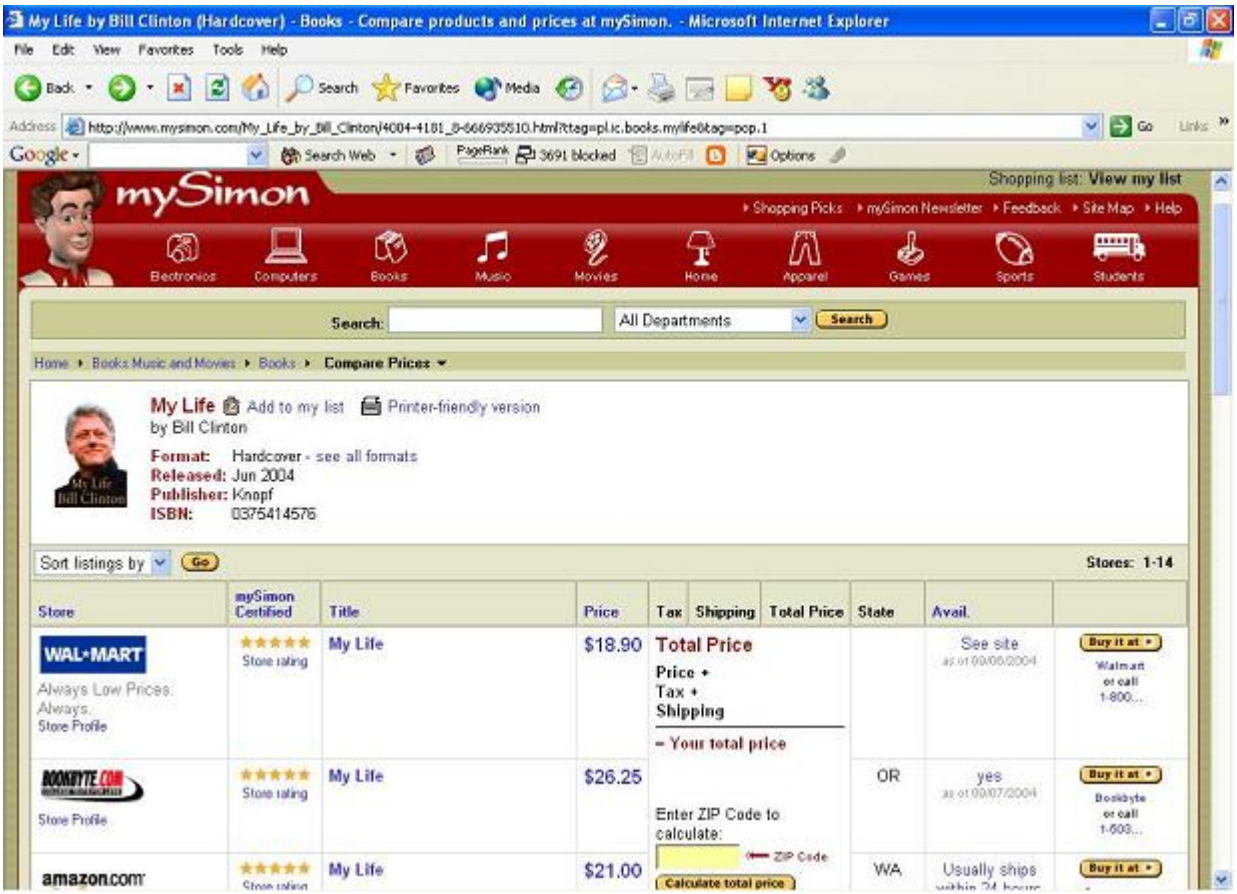

Figure 1. Search Results at mySimon.com

Design of shopbot systems presents an interesting distributed Information Retrieval (IR) challenge. A key challenge in IR is to determine the optimal operational decisions during a distributed IR task based on models of consumer preferences. Specifically in a distributed search application, how can the system determine which data sources to query, how long to wait for responses, and which retrieved results to display in order to maximize consumer utility. In response to a product query by a consumer, a shopbot may search a number of distributed databases (for example, websites of different online stores that may carry the product) or a single database (as in the case of a travel service such as Expedia and a number of other shopbots such as Froogle that cache the distributed content locally). The shopbot may retrieve over 30 (sometimes over hundred) results and typically displays all the retrieved results. Each result (or offer) may contain a number of attributes such as price, airline name, departure time, number of hops, etc that a consumer must evaluate. Key operational decisions that shopbots need to make during the search include which stores to query, how long to wait for responses and which offers to display.

These decisions are critical because they impact the consumer's wait time and cognitive load in evaluating alternatives. Various studies (Kehoe et al 1999) have shown that consumers have significant costs associated with waiting. Thus, shopbots need to trade-off the benefits from waiting (retrieving more offers) against the costs. In addition, most shopbots present all possible results, several of which are dominated or redundant. Johnson and Payne (1984) show that consumers experience significant cognitive costs when evaluating a large set of alternatives. In this paper, we focus on the operational decisions made by a shopbot - which stores to query for offers, how long to wait for these stores to respond, and which items to report to a user – and explore how the operational decision-making may be modified to reduce the costs. In Montgomery, Hosanagar, Krishnan, and Clay (2004), we developed closed-form analytic solutions to the problem. However, the assumptions required to derive the solutions were quite restrictive.

Furthermore, the analysis generated a number of useful comparative statics but not an operational algorithm. In this paper, we use a computational and numerical optimization-based approach that applies to general cases as an alternative to closed-form analytic results for special cases. The objective is to create an algorithmic approach that can be used for decision-making in shopbots.

2. Prior Work

A rich stream of literature in the Information Retrieval (IR) area has studied the design of efficient and effective IR systems. The field has been highly interdisciplinary, drawing from library and information sciences, computer science, and statistics. Some areas of interest include IR models for locating and ranking relevant documents, distributed IR, human interaction, filtering, clustering, question answering, and multimedia IR. IR models are evaluated against two important metrics – precision and recall. Precision measures the fraction of retrieved documents that are relevant and recall measures the fraction of relevant documents that are retrieved. Vector space models [Salton 1975] use a vector representation of documents and queries, and use a cosine measure to determine similarity between documents and queries. Probabilistic models [Fuhr 1992; Ponte and Croft 1998] compute similarity measures based on the probabilities of relevance which are updated using Bayes rule. Alternatively, Chakrabarti et al [1999] discuss the use of hyperlinks on the web to rank order retrieved results (an approach also used by Google). A number of other techniques based on Boolean models, neural networks, etc are also used. Distributed IR [Lu and Callan 2003] is concerned with the challenges of retrieval from distributed heterogeneous data sources. Human interaction issues include user behavior [Kelly and Cool 2002], personalization [Dumais et al 2003] and predicting user access patterns [Chen et al 2002].

In addition, a rich stream of literature has studied software agents for information search and filtering. Software agents are programs that can act independently and autonomously to complete tasks on behalf of users. A key feature of a software agent is its ability to act

autonomously based on information gathered from its environment. Research on agent application in information search and filtering include studies of agent adaptation [Doorenbos et al. 1997], proactive search [Rhodes and Maes 2000], and common sense inference of search goals by agents [Liu et al 2002].

Given that there are techniques to query multiple resources and to determine relevance scores for retrieved documents, this research highlights an approach for developing an algorithm for operational decision making based on models of consumer utility. Prior work in marketing has shown that consumer behavior is an important factor in information search functions. Building on this literature, Montgomery et al (2004) integrate computational considerations with user behavior considerations. The notion of cognitive overload has a long history in consumer behavior research [Jacoby 1984, Keller and Staelin 1987]. Johnson and Payne [1985] show that consumers are willing to trade off cognitive effort in the decision-making process for information accuracy. Montgomery et al (2004) directly incorporate measures of the cognitive costs of decision making [Shugan 1980] to offset the benefit of more information. In addition, models of consumer utility help search agents evaluate consumer preferences for different attributes of information to be displayed. In the management science area, Smith and Brynjolffson [2001] have studied consumer decision making at shopbots and evaluated how consumers trade off different attributes such as product price, delivery time, store brand, etc.

3. Decision Making Framework

Figure 2 illustrates the framework we propose for the shopbot's operational decisions. First, the shopbot is given a product, e.g. a book, to search. Second, the shopbot determines which stores to search and how long to wait for a response. This decision is based on prior (offline) analysis of the response time of stores and the price of that book at various stores (or the category that the book belongs to – for example, hardcover bestseller, etc). Although the wait time is pre-computed, we allow

for the shopbot to make a decision in real time on whether to wait a little longer, based on current retrievals and estimates of response times and prices from past retrievals. Finally, the shopbot decides which of these offers to present to the customer.

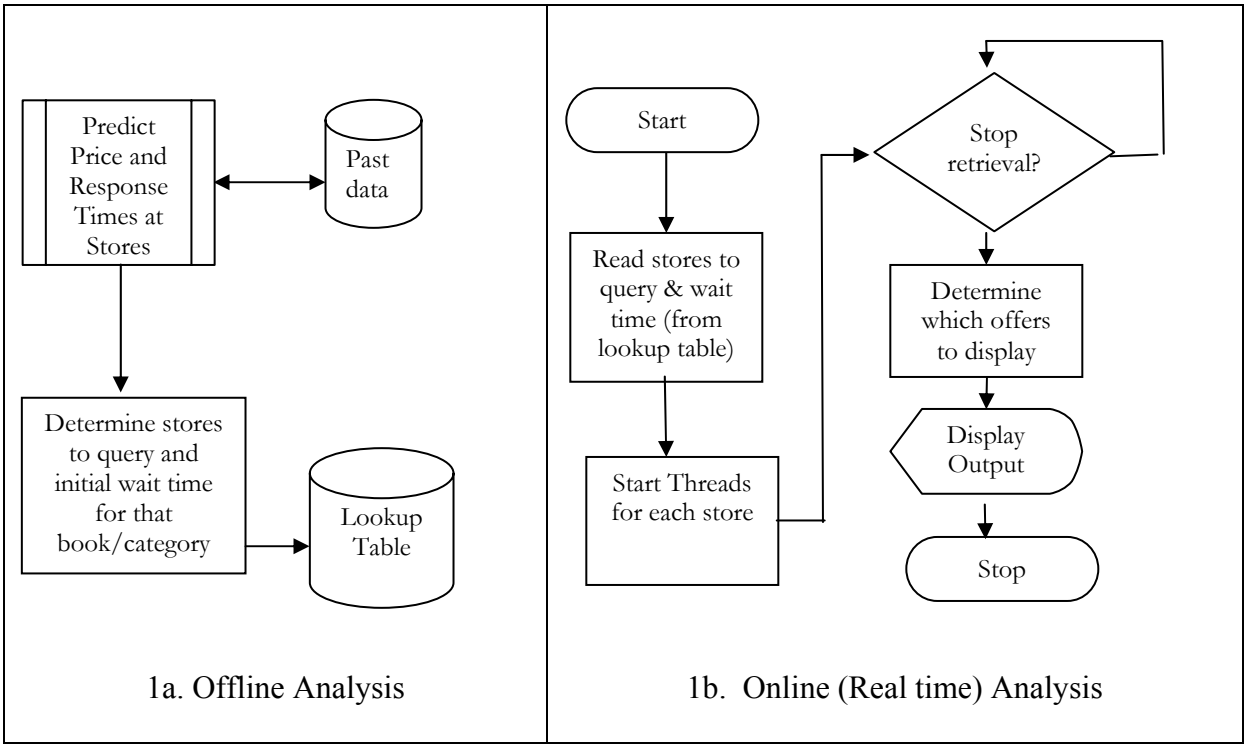

Figure 2. Decision process for a Shopbot

The brief model presented here is based on the one introduced in Montgomery, Hosanagar, Krishnan, and Clay (2004). Let S be the total number of stores that can be queried and the vector q be the stores queried, with q_i=1 if store i is queried and q_i=0 otherwise. T denotes the shopbot wait time and the vector r records the stores retrieved, that is r_i=1 if store i responds within T seconds and r_i=0 otherwise. Finally, the vector p represents the offers presented, i.e., p_i=1 if offer i is displayed and p_i=0 otherwise. The total number of stores queried, stores retrieved, and offers presented equal Q, R, and P, respectively ($Q=\sum q_i$, $R=\sum r_i$, $P=\sum p_i$). Note also that an offer can be displayed only if it has been retrieved, i.e. p_i=1 only if r_i=1. In other words, we do not consider the possibility of a shopbot displaying cached results for stores whose offers were not retrieved in real time.

The utility from an offer i is U_i and is denoted as a sum of a deterministic ($\overline{U_i}$) and stochastic component (ε_i): $U_i = \overline{U_i} + \varepsilon_i$. For example, the utility from an offer may be $U_i = \beta_1.price + \beta_2.shipping\_time + ... + \varepsilon_i$. The deterministic component is a weighted sum of the attributes that make up a product such as price, store rating, tax, and shipping time. The error term represents the uncertainty associated with predicting utility. We assume that these ε_i are independently and identically distributed random variables and follow an extreme value distribution with a zero location parameter and scale parameter of 1. The choice of the extreme value distribution is motivated by its extensive use in choice models (McFadden 1980).

The consumer's expected utility from a query at a shopbot is given by:

$$E[U] = E[\max(U_{p[1]}, U_{p[2]}, ..., U_{p[P]}) - \xi T - \lambda(A-1)(P-1)] \qquad (1)$$

Since the consumer will choose only one item within a set of offers, the utility from a set of choices is equal to the utility from the best alternative or the maximum of the set. Thus, the utility to the consumer from a set of P offers (numbered *p[1], p[2],..., p[P]*) is $\max(U_{p[1]}, U_{p[2]}, ..., U_{p[P]})$. ξ denotes the consumer's disutility of waiting 1 second and T is the shopbot wait time. Shugan (1980) proposed the metric $C = \lambda(A-1)(P-1)$ for the cognitive cost associated with comparing *P* alternatives, each with *A* attributes. Higher the number of items (*P*) or the attributes (*A*) that must be compared, the higher is the cognitive cost. The shopbot's decision problem is:

$$\max_{q,p,T} \{E[U]\} \text{ s.t. } p \le r \le q \qquad (2)$$

The constraint indicates that only stores that are queried can be retrieved, and only offers that are retrieved can be presented. The optimization problem is a stochastic integer programming problem with a very large decision space. In total, there are $\sum_{i=1}^{S}\sum_{j=1}^{i}\left(\frac{S}{i}\right)\left(\frac{i}{j}\right)$ possible sets that the shopbot

could display (ignoring query termination, otherwise the set is larger). A universe of 30 stores yields more than 205 trillion combinations. At the time of our data collection in 2000, shopbots typically operated at q_i =1 $\forall$ i, T=30 seconds, and p=r. That is, they queried all stores, and displayed all offers retrieved within 30 seconds. Over the years, store response times have improved and shopbot wait times have considerably reduced.

Developing an algorithm to determine the stores to query, wait time and offers to present poses unique challenges. These challenges arise from two unique properties of the problem:

i. *Uncertainty*: At the time the shopbot decides on how long to wait for responses, the price of the product at the different stores and the store response times are not known. The shopbot may decide to terminate a search but a very good offer may have been retrieved half a second later. Alternatively, the shopbot may choose to wait for the store's response, but find that the store ends up taking too long to respond. Or the store may in fact respond soon but the price may end up being much higher than anticipated. The uncertainty with respect to store prices and response times thus represents a non-trivial challenge.

ii. *Real-Time nature*: Since a user is waiting for responses, the shopbot should be able to make these decisions very fast. This real-time requirement guides our decision to search for an algorithm, which takes very little time to run and may accept a solution that is not necessarily a true optimum (but is close enough).

4. Characterizing the Optimum

We first note from equation 1 that the number of stores queried imposes no negative externality on the user. Even if the bot queries a large number of stores, it still chooses the optimal query termination time and presentation set. Thus, the shopbot will find it optimal to query all stores. Note that the shopbot may have server capacity constraints that may cause it to economize on number of stores queried even though it reduces consumer utility. The impact of capacity

constraints is analyzed in Section 5. In this section, we focus on a utility-maximizing shopbot and restrict our attention to the remaining two decisions.

The shopbot thus needs to determine how long to wait and which offers to present. We solve this problem by looking at these 2 decisions sequentially. The merit of this approach lies in the observation that at the time of presentation, all the information about store attributes (price, etc) is known (i.e., $\overline{U}_i$ is deterministic). However, when the shopbot needs to determine how long to wait, these variables can at best be estimated and are thus stochastic.

At the time of presenting the offers, $\overline{U}_i$ is known for all stores. There is still uncertainty associated with predicting utility denoted by the error terms ε_i. We can obtain the expectation of the highest order statistic to get $E[\max(U_{p[1]}, U_{p[2]}, ..., U_{p[P]}) = \ln(\sum_{i=1}^{P} \exp\{\overline{U}_{p[i]}\}) + \gamma$, where γ is the Euler's constant []. This gives us:

$$E[U \mid P, R] = \ln(\sum_{i=1}^{P} \exp\{\overline{U}_{p[i]}\}) + \gamma - \xi T - \lambda(A-1)(P-1) \quad (3)$$

To find the set of *P* offers that maximize the expected utility, we first sort the retrieved offers by their expected values, $\overline{U}_i$. The offers with the highest expected utility are also the best offers to present. This follows from the i.i.d. assumption of ε_i because the likelihood that the actual utility from an offer may be higher (or lower) is equal for all offers. The conclusion can also be verified from equation 3 which shows that given the number of offers to display, *P*, it is best to choose the *P* offers with the highest expected values in order to maximize $E[U \mid P, R]$. This implies that we do not need to evaluate all the 2^R combinations but it suffices to look at the *R* sets with the top 1, 2,…*R* offers respectively. This observation serves to increase the computational efficiency of our algorithm presented in Figure 3. It is obtained by computing the expected utility from all the R combinations and choosing the one that provides the maximum utility. The

computational complexity of the algorithm is O(R.logR). Sorting the offers using quick sort takes O(R.logR) time. The while loop takes O(R) and thus the entire algorithm is O(R.logR).

1. Input offer-1, offer-2,…… offer-R (Set of 'R' retrieved offers)
2. Sort the R retrieved offers in decreasing order of $\overline{U}$.
3. Set Optimum_Utility to Utility of offer-1
4. Set Optimum_P to 1
5. Set i to 2
6. While i <= R do
7. Set Util to Utility of the set of top 'i' offers
8. If Util > Optimum Utility
9. Set Optimum_Utility to Util
10. Set Optimum_P to i
11. Endif
12. Set i to i + 1
13. Endwhile
14. Output Optimum_P and Optimum_Utility
15. Halt

Figure 3. Algorithm for determining the presentation set.

At the time the shopbot decides on the wait time, neither $\overline{U}_i$ nor the response time of any store is known. The shopbot has to determine a T* such that any potential improvement in the utility of the offers obtained by waiting more is outweighed by the cost of waiting. The marginal cost of waiting is known (ξ). However, it is difficult to ascertain the marginal benefit of waiting an additional second due to the highly non-linear and stochastic nature of the impact of T on the utility from the offers (see equations 1 and 3). This is because *r* varies with T in a stochastic way and further *p* varies with *r* (in a deterministic way). The shopbot's decision problem is:

$$\max_T \sum_r \Pr(r).E\left\{\ln\left(\sum_{i=1}^{P}\exp\{\overline{U}_{R-i+1:R}\}\right)+\gamma-\xi T-\lambda(A-1)(P-1)\mid r\right\}$$

That is, we need to multiply the probability of retrieving a vector *r* (given a choice of T)

with the expected utility of the associated optimal presentation set and sum it up for all the 2^S combinations of *r*. 30 stores have over 10^9 combinations. This needs to be evaluated for different values of T as we move towards the optimal T*. That is clearly not efficient.

In order to develop an algorithm to determine T*, we run simulations to obtain the properties of the true global optimum and explore algorithms that can provide solutions that are close enough. April et al. (2001) and Glover et al. (1999) provide a useful primer on the merits of combining simulation and optimization in managing the complexity and uncertainty posed by many real-world problems.

We simulate store prices and response times based on real empirical data and analyze the optimal wait time for the shopbot. To illustrate the procedure, we consider the case of 60 books from the New York Times bestsellers list. We constructed automated agents and collected prices and delivery information at 30 online bookstores. In this paper, we use a six-month panel from August 1999 to January 2000 for the 60 books. Table 1 presents the mean and standard deviation of book prices (normalized to the list price of the book[1]) at the various stores. The prices of the books are normally distributed at each store. Thus, with no additional information except that the book is a New York Times bestseller, the shopbot's best estimate is that Buy.com has the lowest price at 52% of the list price. The shopbot can generate such information in the form of lookup tables for different product categories.

Store	Mean	Std. Dev.	Store	Mean	Std. Dev.
1bookstreet	0.76	0.13	Buy.com	0.52	0.10
A1 Books	0.75	0.06	Cherryvalleybooks	0.89	0.02
alldirect.com	0.63	0.05	Christianbook	0.74	0.06
AlphaCraze.com	0.64	0.09	Classbook.com	0.96	0.06
Amazon	0.63	0.13	Codys Books	0.99	0.06
Baker'sDozen online	0.99	0.06	computerlibrary.com	0.99	0.06

[1] That is, for each book we used (Price of Book at Store)/(List Price of Book) to make comparisons across books and computation of means meaningful.

Barnesandnoble.com	0.63	0.13	Fatbrain	0.65	0.15
BCY Book Loft	0.72	0.07	HamiltonBook.com	0.70	0.07
bigwords.com	0.77	0.06	kingbooks.com	0.73	0.04
Book Nook Inc.	0.99	0.05	page1book.com	0.99	0.07
Bookbuyer's Outlet	0.62	0.13	Powells	0.91	0.04
Books.com	0.70	0.09	Rainy Day Books	0.89	0.05
booksamillion.com	0.59	0.12	Rutherfords	0.89	0.05
booksnow.com	0.88	0.06	varsitybooks.com	0.75	0.05
Borders.com	0.62	0.13	WordsWorth	0.83	0.10

Table 1. The mean and standard deviation of book prices (normalized by list price)

Similarly, we recorded the response time of the stores for the queries. We found that almost 80% of requests were retrieved in under 2 seconds. However, network congestion or server overload can cause substantial delays or even no response. We model that stores do not respond with a probability p and their response times follow a gamma distribution when they do respond (with probability 1-p). An estimate of the probability of response and the gamma parameters for three major stores are given in Table 2. We found that the fit is good for our data. It must be noted that the parameters will have changed over the last five years in a direction to reflect improved server and bandwidth capacity in the network.

Store	No. of obs.	1-p	Gamma Parameters		Moments	
			Location (α)	Scale (σ)	Mean	StdDev
BCY Bookloft	7,803	.979	.452	5.61	2.53	3.77
Amazon	5,739	.960	.775	3.47	2.69	3.05
Barnes-Noble	2,224	.950	.443	5.94	2.63	3.95

Table 2. Estimates of gamma distribution and probability of response for selected bookstores.

In the simulations, we consider a book with a list price of $30. The price of the book at the stores and the store response times are drawn from random distributions with parameters as in Table 1 and Table 2 respectively. In order to compute the utilities, we have to first determine the weights for the utility function. These weights can be estimated directly from previous purchases at the shopbot or be specified by the user. We use the maximum likelihood estimates reported by Brynjolfson and Smith (2000). Their estimates are based on an analysis of consumer choice at

Dealtime (a shopbot that has since been renamed to Shopping.com) and are listed in Table 3. The table indicates that if the item price of a book increases by 1$, the utility drops by 0.194 utils and similarly, if the book is ordered from Amazon, the utility increases by 0.477. More intuitively, the average shopbot user is willing to pay $0.477/0.194 = \$2.46$ more to Amazon than a lesser branded retailer. Further, we assume that the value of time is \$.01/second, i.e., $\xi = 0.01 * 0.194 \approx 0.002$ utils. A justification for this time cost is that if a user makes \$70,000 annually, each second is roughly worth \$.01. Finally, we assume that a consumer can make one comparison a second but considers it five times as costly as waiting due to the cognitive effort expended i.e., $\lambda = \$0.05 = 0.01$ utils.

Parameter	Item Price	Shipping Cost	Expected Days to delivery	Amazon	BarnesandNoble	Borders
Estimate	-0.194	-0.368	-0.019	0.477	0.177	0.266

Table 3. Utility Weights used in the simulations

For these settings, we conducted 100,000 simulations and determined the wait time that would have maximized consumer utility. We do not consider how the shopbot would obtain this optimum value but instead consider ex-post (with the knowledge of prices and response times) what choice of T* would have maximized the utility. That is, if there was an oracle that could have precisely computed the prices and response times (and eliminated the stochastic component), then the shopbot could compute the utility associated with any choice of T*. The purpose of these simulations is to characterize the true optimum and explore how we can get close enough. To evaluate the optimum wait time, we only need to evaluate the expected utility at 30 values of T (the response times of the 30 stores). To see why this is so, consider any wait time T that lies between the response times of 2 stores t_i and t_{i+1}. With this choice of T, all stores from 1…i are retrieved and the remaining stores are not retrieved (the stores are assumed to be sorted in

increasing order of response time). If the shopbot reduces the wait time to t_i instead, the same stores are retrieved but the wait time is reduced. This increases the utility to the consumer. Thus, the candidate values of T that would maximize the expected utility are $t_1, t_2, \ldots t_S$. Thus, it suffices to evaluate the utility at S discrete values of T alone. This makes the problem of determining the optimal T^* tractable. The results of the simulations are summarized in Table 4. The median optimal waiting time was 6.13 seconds, with the 90th and 95th percentile occurring at 8.48s and 9.28s respectively. Thus, in 95% of the cases, it sufficed to wait for less than 9.28 seconds. We expect that the min, max and average values of optimal wait times have all shifted downwards as servers have gotten faster over the years. This is an indication that shopbots that wait for 20-30 seconds are clearly operating at sub-optimal levels. The number of offers presented varies from 1 to 20 with a mean of 8.81. This highlights that it is important to evaluate what offers to display and eliminate alternatives that are dominated.

	Min	Max	Average	Std. Dev.
Optimal Wait Time (sec)	2.02	18.19	6.33	1.61
Utility	-3.42	-1.39	-2.69	0.21
Number of Offers Presented	1	20	8.81	3.93

Table 4 Summary of simulation results

4 Performance Analysis

The results of section 3 clearly indicate that optimal wait times for shopbots were much lower than their chosen wait times at the time of data collection. However, the shopbot still faces the issue of determining the optimal wait time because the previous simulation identified the global optima by considering the response times and prices to be known. To evaluate how shopbots might develop useful heuristics for determining the wait time in real-time searches, we simulate searches at a shopbot and compare three different algorithms. We continue to look at the

case of a New York Times bestseller with list price of $30 and a user with utility weights as in section 3. We consider 3 alternative algorithms – 1) *current shopbots* which query all stores, wait for 30 sec and display all results that are retrieved; 2) a *static shopbot* that waits for 6.33 seconds (average optimal wait time in the previous section; and 3) An *adaptive shopbot* that initially sets a wait time of 6.33 seconds and simultaneously retrieves the table of price predictions and computes expected utility. If the 10 stores with the highest expected utility are retrieved before 6.33s, it ends the search prematurely. If 6.33 seconds have elapsed and any of the top 3 offers have not yet been retrieved, it resets the wait time to 9.28 sec (95^{th} percentile in previous simulations). If at any time during this additional wait, it has the top 3 offers, the shopbot ends the search right away, else it waits for a total of 9.28 seconds. Clearly, none of these algorithms will be truly optimal but the question is whether simple heuristics can provide solutions that are close enough. For the presentation decision, we use the algorithm presented in Figure 2.

Figure 4 shows the results of ten simulations[2]. Both static and adaptive shopbots perform better than current shopbots, with adaptive shopbots dominating the performance of static ones. The summary of the results from the 100,000 simulations is displayed in table 5. On average, the static shopbot provides the consumer with a utility gain of 0.30 over current shopbots, which is equivalent to $1.55 (a dollar increase in item price reduces consumer utility by 0.194 utils, so 0.30 utils $\approx$ $1.55). On average, the shopbot consumer pays $17.1 for a bestseller with a list price of $30. Thus, relative to the price paid by the consumer, the increase in utility is 1.55/17.1 = 9.04%. Similarly, usage of the adaptive shopbot is associated with an increase in utility of 0.32 utils on average. This is equivalent to $1.65 or 9.65% relative to the price paid. These gains are non-trivial and can be a source of differentiation for shopbots. Notice also that the average utility from the adaptive shopbot is -2.70 whereas the average utility in section 4 was -2.69. Thus, on average the

[2] Although 100,000 simulations were conducted, 10 simulations are graphed for simplicity and ease of visualization.

adaptive bot comes within 0.01 utils of the global optimum possible and is clearly an acceptable algorithm to use. Such an algorithm was easily obtained by analyzing the simulation results and subsequently developing appropriate heuristics. Note also that the algorithm can easily accommodate a learning component by updating the lookup table, which stores the statistics regarding optimal wait time, using new data (possibly weighting new data more).

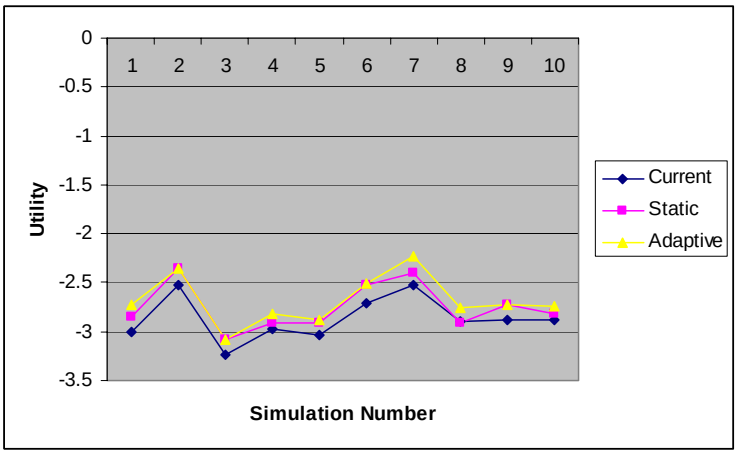

Figure 4. Consumer Utility in the first ten simulations

Shopbot	Average Utility	Min	Max	$ Value of Gain (Avg)
Current	-3.02	-3.76	-1.96	-
Static	-2.72	-3.58	-1.39	1.55
Adaptive	-2.70	-3.43	-1.39	1.65

Table 5. Summary of Performance Analysis

5 Discussion and Conclusions

In this paper, we develop algorithms to enable shopbots to make their operational decisions in an intelligent manner. We demonstrate how shopbots can mine information from previous retrievals to determine the distribution of prices and response times of stores and leverage that to improve operational performance. In particular, we showed how the stochastic and real-time nature of the problem can be addressed by using heuristics that provide solutions close to the optimum. The adaptive algorithm proposed in the paper provides a simple decision rule that can

run in real-time with no overheads and yet provide very good solutions. While the framework was illustrated for the case of New York Times Bestsellers, shopbots can use the approach for any product category.

In this paper, we only considered a shopbot that is interested in maximizing consumer utility without any other constraints. Often, profit sharing arrangements with certain stores may make it sub-optimal for the shopbot to filter results. We have ignored any such profit considerations. In addition, shopbot server constraints may alter its decision to query all stores. Although the framework was illustrated for the case of shopbots, it is applicable to other Information Retrieval systems as well where decisions about which items to retrieve and present to a user must be made.

References

[] April, J., F. Glover, J. Kelly and M. Laguna (2001), "Simulation/Optimization Using "Real-World" Applications," *Proceedings of the 2001 Winter Simulation Conference*.

[] Brynjolfsson, Erik and Michael D. Smith (2000), "The Great Equalizer? Consumer Choice Behavior at Internet Shopbots," CMU Working Paper, Heinz School, Pittsburgh, PA.

[] Glover, Fred, J. Kelly and M. Laguna. "New Advances for Wedding Optimization and Simulation," *Proceedings of the 1999 Winter Simulation Conference*.

[] Johnson, Eric J. and J. W. Payne (1985), "Effort and accuracy in choice", *Management Science*, 31, pp. 394-414.

[] Kehoe, Colleen, J. Pitkow, K. Sutton, G. Agarwal and J. D. Rogers (1999), "Results of the Graphics, Visualization and Usability Tenth World Wide Web User Survey," Working Paper.

[] McFadden, D. (1980), "Econometric Models of Probabilistic Choice Among Products", *Journal of Business.*

[] Montgomery, Alan, K. Hosanaagr, R. Krishnan and K. Clay (2004), "Designing a Better Shopbot," *Management Science*, Vol 50, No. 2.

[] Shugan, Steven M. (1980), "The Cost of Thinking", *Journal of Consumer Research*, Vol. 7, September, pp. 99-111.