Resource-Bounded Searches in an Information Marketplace

VICTOR LESSER, BRYAN HORLING, ANITA RAJA, AND XIAOQIN ZHANG
University of Massachusetts at Amherst
THOMAS WAGNER
University of Maine

BIG is an information-gathering agent that processes Web documents to create product models and recommend purchases based on user selection criteria. New capabilities support sophisticated controls on the money spent to acquire information from sites that charge an access fee.

Advances in information retrieval (IR) technologies have led to the development of such tools as AltaVista and Google, which give information seekers a starting point for their searches. However, in many cases, manual browsing through even a limited portion of the relevant information is no longer effective. The problem originates not in the IR technologies, but rather in the volumes of information available via the Web and the generality of the term-frequency approach to searches. For any given query, there are often simply too many relevant documents for the human client to process the information efficiently.

Over the past five years, we have been developing a Web-based, resource-bounded, information-gathering agent—called BIG, for Bounded Information Gathering—to support a human decision process. Although its techniques are general enough to apply to a wide range of domains, BIG specifically helps clients pick software packages. For example, a client can instruct BIG to recommend a database package for Windows 98 and specify constraints on both the search process and the product characteristics. BIG will then formulate a plan, locate and extract relevant information from both structured and unstructured documents, and return a recommendation to the client, along with supporting data.

The BIG system and its performance are well documented.[1,2] We give a brief overview of how it works in the sidebar on p. 51, "BIG Agent Architecture and Information-Gathering Scenario," but our focus here is on a recently developed capability, namely, to schedule information-gathering activity in a way that controls the money spent on acquiring information from sites that charge a fee for access. This capability supports an "information marketplace" on the Web, which we feel will eventually supersede (but obviously not completely replace) the current model of free information access supported by advertising fees.

1089-7801/00/$10.00 ©2000 IEEE

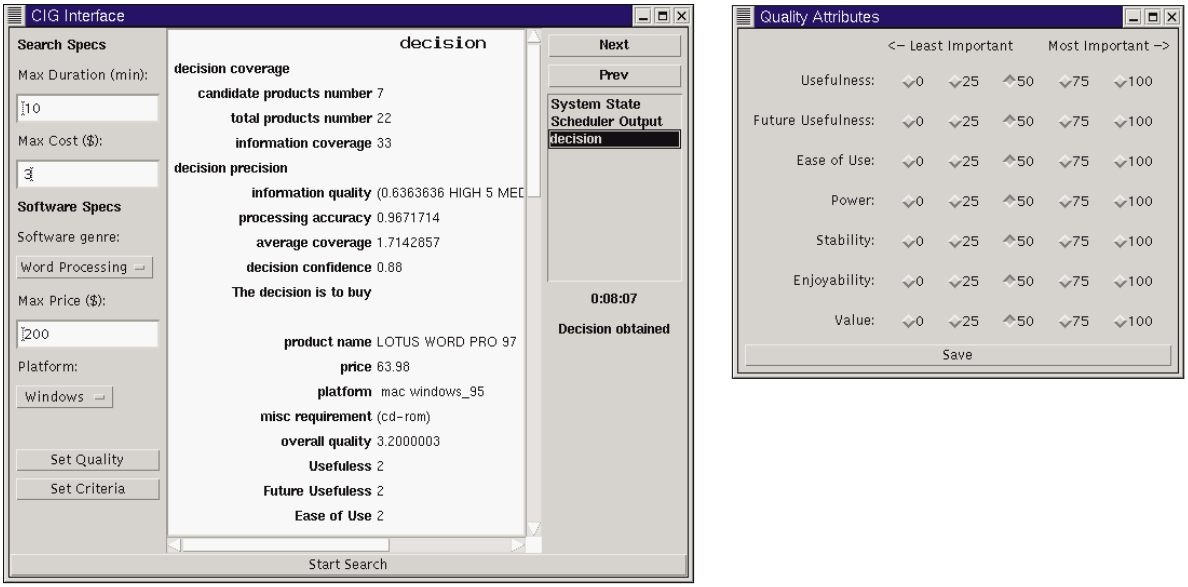

Figure 1. User interface (on the left) for a query-response episode using the BIG resource-bounded information-gathering agent, and (on the right) for weighting quality attributes.

In fact, a number of sites already charge access fees for information that is (theoretically) screened for quality. Such information could be especially useful to an intelligent agent in systems like BIG, which support purchasing and other high-level decisions. These decisions require extensive information gathering and high confidence in the extracted information that make the potential increased accuracy worth paying for.

Recent work on digital libraries supports this view of an information economy.[3] Though the work we report here does not emphasize a marketplace model for negotiating the costs of information acquisition, our basic approach will support this more dynamic costing model.

INFORMATION GATHERING ON THE WEB

This new capability fits very nicely with BIG's existing ability to control both the end-to-end time to gather information and the balance between the information's scope/coverage and the resulting decision's precision. From the perspective of the software domain, this balance refers to the number of products discovered and analyzed versus the amount of detailed and overlapping information gathered on each product and used to make the final decision.

More generally, BIG is designed with the view that information gathering on the Web must be resource-bounded and that different degrees and dimensions of resource boundedness are appropriate in different situations. Figure 1 shows the user interface for a typical query/response search episode. The client's initial query directed BIG to search for a word-processing package for Windows; the search process itself was to take no more than 10 minutes and cost no more than $3. The user could set additional preferences in the quality attributes window shown at the bottom of the figure, but in this case has assigned equal weight to all preferences. The right panel in the main screen shows that a decision was returned to the user after approximately eight minutes; the center panel shows some of BIG's characterization of both the selected product and the overall decision process.

Much of BIG's strength is derived from its use of AI mechanisms to dynamically integrate a variety of information-retrieval and -gathering techniques.

- Like a metasearch engine (for example, Zurfrider at http://www.zurf.com/), BIG may use multiple Web search tools to locate information.[4,5] Unlike the metasearch engines, BIG learns about products over time and reasons about the time, cost and quality trade-offs of different Web search options.
- Like a personal information agent (for example, Robo surfer at http://www.robosurfer.com/), BIG actively gathers documents on the Web by using search engines and following chains of

BIG Agent Architecture and Information-Gathering Scenario

Figure A shows the major BIG agent architecture components:

- *Resun planner*—a blackboard-based interpretation planner that is the domain expert and driving force behind the information-gathering and decision-making process (from REsolving Sources of UNcertainty).
- *Information extractors*—text-extraction tools use a variety of techniques to process free-format text, converting it into structured data.
- *Document classifiers*—text-processing filters quickly qualify text before sending it on to the information extractors, thus eliminating unrelated documents before more complex techniques are used on them.
- *Server information database*—a local database of information sources stores qualitative data, such as average server response time and cost, which are used in generating information-gathering plans.
- *Object database*—a local database stores product information during the construction of product models, including information learned from prior searches.
- *Design-to-Criteria (DTC) scheduler*—an agent-control problem solver enables BIG to meet deadlines and cost restrictions by selecting an appropriate course of action from a problem-solving model.
- *TAEMS modeling language*—a Task Analysis, Environment Modeling, and Simulation language is used to quantify and model an agent's problem-solving behaviors for the DTC scheduler.
- *Task assessor*—a software module manages the interface between the Resun opportunistic planner and the DTC scheduler, converting the problem-solving process generated by Resun into a TAEMS structure usable by DTC.

The control and information flow in a BIG scenario begins with input through the user interface to the Resun planner, which formulates a rough description of the information needed to generate a useful decision. The task assessor uses these information requirements to generate a TAEMS model,[1,2] which organizes and quantifies a set of high-level information-gathering plans.

The DTC scheduler decides which high-level plan is most appropriate and how best to plan activities based on user-specified performance criteria. Resun uses the schedule to begin information retrieval, culling data from online databases, search engines, or static Web pages. Document classifiers quickly categorize and rate the resulting information; documents passing this phase can then be processed by information extractors, which attempt to distill the data into a concise record.

The interpreted data is used to generate product models, which are incrementally improved over time. At the end of the process, the user's product specification input is used to select the most appropriate product from the set generated on the blackboard.

References

1. K. Decker and V. Lesser, "Quantitative Modeling of Complex Computational Task Environments," *Proc. 11th Nat'l Conf. Artificial Intelligence*, AAAI/MIT Press, Cambridge, Mass., 1993, pp. 217-224.
2. B. Horling et al., "The TAEMS White Paper," 1999; available online at http://mas.cs.umass.edu/research/taems/white/.

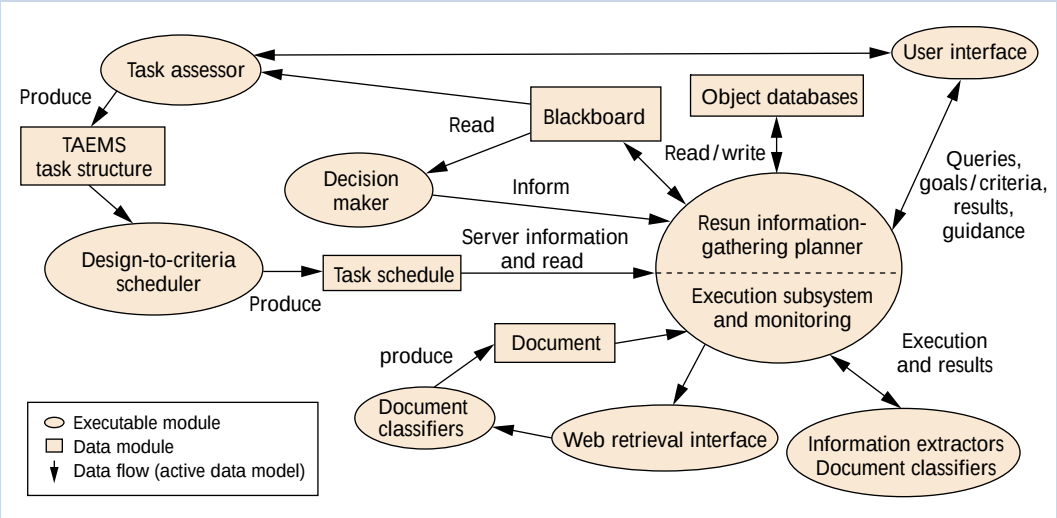

Figure A. The BIG agent architecture includes seven main components for planning, scheduling, informational retrieval, text processing, knowledge management, and decision making.

links; however, in addition to locating relevant information, BIG analyzes the documents using a variety of techniques that include natural language text-extraction and site-specific wrapper utilities. Conceptually, BIG "reads" free-format text, identifies product features like price, disk requirements, and support policies, extracts these features from the documents, and then reasons about them.

- Like shopping agents (for example, Jango at http://www.jango.com), BIG gathers information to support a decision process.[6] However, BIG differs in its complex decision process and its runtime information-processing facilities.
- BIG is related to the Warren[7] multiagent portfolio management system, which also retrieves and processes information, but differs in several areas: its reasoning about the trade-offs among alternative ways to gather information, its use of gathered information to drive further gathering activities, its bottom-up and top-down directed processing, and its explicit representation of sources of uncertainty associated with both inferred and extracted information.

In the remainder of this article, we discuss the general framework that allows BIG to make resource-bounded decisions that organize its processing to achieve cost objectives. Then we present detailed experimental data indicating how BIG reorganizes its search process to live within specific monetary constraints on how much can be spent on accessing information useful in making a software purchasing decision.

PLANNING FOR RESOURCE BOUNDS

BIG addresses time and cost limitations by using a domain-independent agent scheduler, the Design-to-Criteria scheduler.[8] The expertise of BIG's DTC scheduler lies in analyzing an agent's candidate set of problem-solving actions and choosing a course of action for the agent that meets complex, multidimensional design criteria. In BIG, the design criteria specify the relative importance of solution completeness and coverage, as well as the restrictions on cost and time. DTC achieves domain independence through the Task Analysis, Environment Modeling, and Simulation (TAEMS) task-modeling framework,[1,2] which describes and quantifies key activities and decision points in the agent's problem-solving process.

The DTC scheduling problem has commonalities with both classic scheduling and planning problems. One aspect of scheduling a TAEMS task structure is deciding which alternative tasks to perform or which methods to use to achieve a particular objective; the other aspect is determining the best sequence in which to perform the activities. We not only order the activities; we can concurrently schedule nonlocal activities. For example, new document-retrieval requests to Web search engines can be scheduled simultaneously with the local processing of documents already retrieved.

This element of choice in BIG's problem-solving process gives DTC the room to maneuver and address resource limitations. Without this flexibility, BIG's problem-solving behavior would not be adjustable to different resource situations.

TAEMS is a domain-independent task-modeling framework used to describe and reason about complex problem-solving processes. TAEMS models are used in multiagent coordination research and are currently deployed in many research projects, including cooperative information gathering, collaborative distributed design, intelligent home environments, and software process coordination. Typically, a problem solver represents domain problem-solving actions in TAEMS, possibly at some level of abstraction, and then passes the TAEMS models on to agent control problem solvers like the DTC scheduler.

TAEMS models are hierarchical abstractions of problem-solving processes that describe alternative ways of accomplishing a desired goal; they represent major tasks and major decision points, interactions between tasks, and resource constraints but they do not describe the intimate details of each primitive action. All primitive actions in TAEMS, called *methods*, are statistically characterized via discrete probability distributions in three dimensions:

- *quality* is a deliberately abstract domain-independent concept that describes the contribution of a particular action to overall problem solving;
- *cost* describes the financial or opportunity cost inherent in performing the action; and
- *duration* describes the amount of time that the action modeled by the method will take to execute.

Uncertainty in each of these dimensions is implicit in the performance characterization. Thus, agents can reason about the certainty of particular actions as well as their quality, cost, and duration trade-offs. Task structure programmers or problem-solver gen-

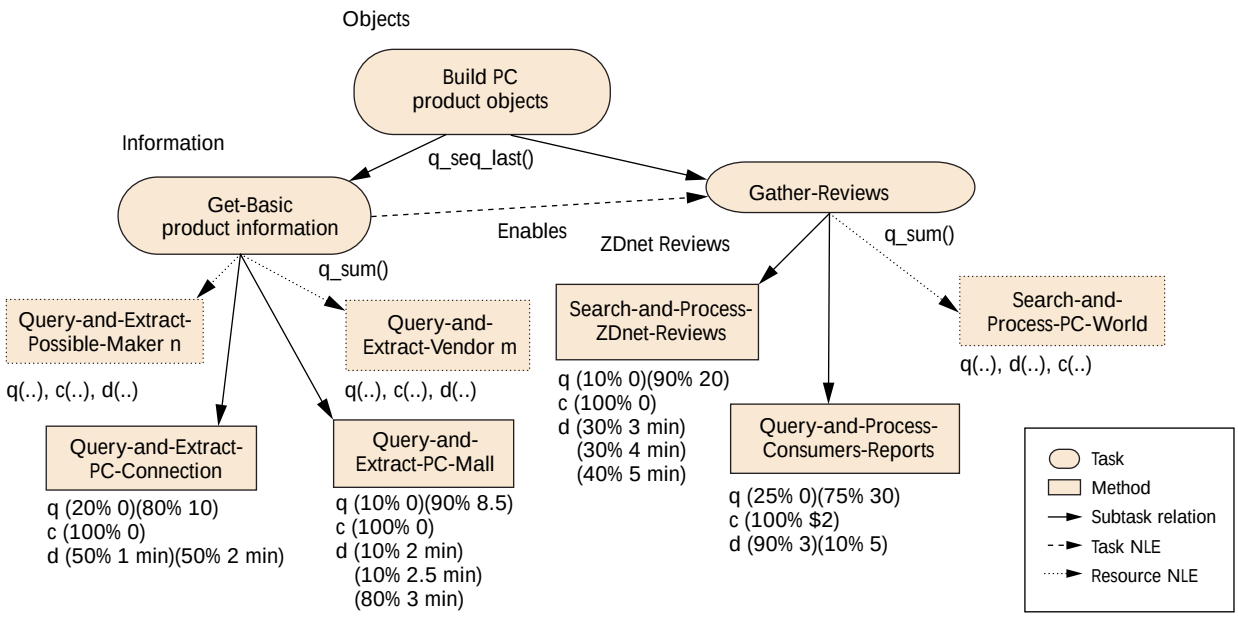

Figure 2. Simplified subgraph of a TAEMS information-gathering task structure. The top-level task is to construct product models of retail PC systems. Task interactions are critical to BIG's behavior model.

erators estimate the performance characteristics of primitive actions, and reasoning components typically replan and reschedule when unexpected events occur.

To illustrate, consider Figure 2. This simplified subgraph of a BIG-generated task structure conceptualizes the portion of the information-gathering process that pertains to constructing models or objects of commercial products. The top-level task is to construct product models of retail PC systems. It has two subtasks, Get-Basic and Gather-Reviews, both of which are decomposed into methods described in terms of their expected quality, cost, and duration.

The Enables arc between Get-Basic and Gather-Reviews is a nonlocal effect (NLE), or task interaction; it models the fact that the review-gathering methods need product names as a precondition to gathering reviews for them. As we will illustrate, task interactions are critical to BIG's behavior model because they describe the relationship between obtaining more documents or more text processing and solution attributes such as coverage or precision.

Returning to the example, Get-Basic has two methods, joined under the sum() quality-accumulation function, which defines how performing the subtasks relates to performing the parent task. In this case, either method or both may be employed to achieve Get-Basic. In general, a TAEMS task structure represents a family of plans, rather than a single plan, where the different paths through the network exhibit different statistical characteristics or trade-offs.

Given the process described in Figure 2, DTC can construct custom schedules for BIG to meet its current situation. Even this simple task structure gives DTC room to adapt BIG's problem solving. Figure 3 (next page) shows four different schedules constructed for different BIG clients that have different objectives and criteria:

- Schedule A is constructed for a client that has both time and financial resources; the user is simply interested in maximizing overall solution quality.
- Schedule B is constructed for a client that wants a free solution.
- Schedule C is constructed for a client interested in trading off quality, duration, and cost equally.
- Schedule D meets the needs of a client interested in maximizing quality while meeting a hard deadline of seven minutes.

Note that schedule D is actually preferred over a schedule that includes method Query-and-Extract-PC-Connection, even though said method has a

Schedule A — Client has no resource limitations; maximize quality.
Query-and-Extract-PC-Connection Query-and-Extract-PC-Mall Query-and-Process-ZDnet Query-and-Process-Consumers-Reports
Quality distribution: (0.02 0.00)(0.00 8.50)(0.00 10.00)(0.02 8.50)(0.04 28.50)(0.02 30.00)(0.18 38.50)(0.01 40.00)
(0.05 48.50)(0.12 58.50) (0.05 60.00)(0.49 68.50)
Expected quality: 55.34 Probability q or greater: 0.66
Cost distribution: (1.00 2.00)
Expected cost: 2.00 Probability c or lower: 1.00
Finish time distribution: (0.09 9.00)(0.35 1.00)(0.03 1.66)(0.47 12.00)(0.05 15.00)
Expected finish time: 11.50 Probability d or lower: 0.45

Schedule B — Client is interested in a free solution.
Query-and-Extract-PC-Connection Query-and-Extract-PC-Mall Query-and-Process-ZDnet
Quality distribution: (0.02 0.00)(0.02 8.50)(0.01 10.00)(0.07 18.50)(0.16 28.50)
(0.07 30.00)(0.65 38.50)
Expected quality: 33.29 Probability q or greater: 0.65
Cost distribution: (1.00 0.00)
Expected cost: 0.00 Probability c or lower: 1.00
Finish time distribution: (0.1 6.00)(0.38 8.00)(0.04 8.45)(0.31 9.00)(0.17 10.00)
Expected finish time: 8.4 Probability d or lower: 0.52

Schedule C — Client requests an even trade-off between quality, cost, and duration.
Query-and-Extract-PC-Connection Query-and-Process-ZDnet
Quality distribution: (0.20 0.00)(0.08 10.00)(0.72 30.00)
Expected quality: 22.40 Probability q or greater: 0.72
Cost distribution: (1.00 0.00)
Expected cost: 0.00 Probability c or lower: 1.00
Finish time distribution: (0.15 4.00)(0.30 5.00)(0.35 6.00)(0.20 7.00)
Expected finish time: 5.60 Probability d or lower: 0.45

Schedule D — Client wishes to maximize quality while meeting a hard deadline of 7 minutes.
Query-and-Extract-PC-Mall Query-and-Process-Consumers-Reports
Quality distribution: (0.10 0.00)(0.29 8.50)(0.61 38.50)
Expected quality: 25.88 Probability q or greater: 0.61
Cost distribution: (1.00 2.00)
Expected cost: 2.00 Probability c or lower: 1.00
Finish time distribution: (0.09 5.00)(0.09 5.50)(0.72 6.00)(0.01 6.05)(0.09 8.00)
Expected finish time: 6.05 Probability d or lower: 0.90

Figure 3. Custom schedules for the information-gathering task structure.

higher expected value than Query-and-Extract-PC-Mall. This is because the PC-Connection method has a higher probability of failure. The Enables NLE from the task of getting product information to retrieving reviews increases the probability of failure and also impacts the probability of being able to query the consumer's site for a review. Thus, though the local choice would be to prefer PC-Connection over PC-Mall for this criteria, the aggregate effects lead to a different decision.

Thus far we have illustrated the DTC planner's capabilities and TAEMS using a small example. Figure 4 shows a complete TAEMS task structure generated by the task assessor.

COST-BASED EXPERIMENTS

To study how BIG addresses cost limitations, we ran a series of experiments. The experimental environment simulated fee-for-access models based on eight websites often used by BIG and arbitrarily assigned different qualities and costs, as shown in Table 1. All other sites are assumed to be free, and their quality is determined by the relative number of external sites that link to them. The assumption is that the more frequently a site is referred to, the higher its quality is likely to be. Methods making queries to these sites are also associated with different quality and cost characterizations, based on the quality and cost associated with the site they access.

Different Schedules Based on Cost

The BIG system constructs different schedules where execution reaches different product decisions based on different cost constraints. In the sample query shown in Figure 1, BIG is to find word-processing software for the Windows platform, given an end-to-end completion time of 10 minutes and a product price limit of $200. The DTC scheduler was used with function or design criteria dictating that it search for a schedule that achieves the high-

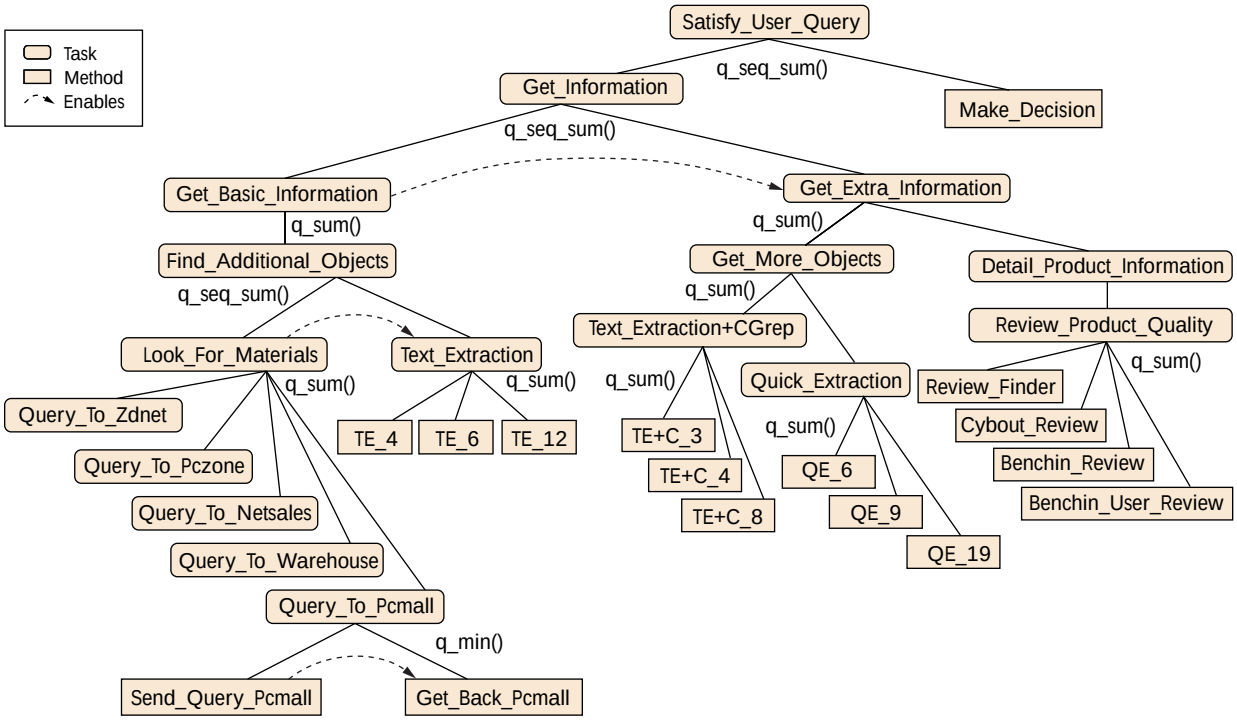

Figure 4. Complete example of BIG information-gathering task structure.

est quality within the cost and time limits. However, meeting cost limits were given a much higher priority than meeting time limits. Thus, the time limit may not be met in every schedule produced by DTC.

Table 2 (next page) shows the different schedules constructed for cost thresholds of $0, $1, $3, $6, and $9. In this table, a "Y" in a row indicates that the method in that row is included in the schedule with a specific cost threshold. For example, method Query_To_Net sales is only used in the schedule where the cost threshold is nine dollars.

When the search cost threshold is $0, the BIG system first makes a query to the free site "zdnet" to get low-quality, basic product information; it then chooses seven text-processing methods to process information (these methods are also free). Next, it elects to query a free review site, "cybout review," to get review information. Finally, it makes a decision based on the available information.

In contrast, when given the search cost threshold of $1, the system makes an extra query to the "pczone" site that charges $0.8 and gains access to medium-quality information. When the search cost threshold increases to $3, the system spends $2 on the "benchin" review site to get high-quality information. Given $6, the system queries three product sites: "zdnet," "pczone," and "pcmall." It also queries the "benchin" site twice for different review information: company reviews and user reviews. With a searching cost of $9, the system chooses to query all five product sites and three review methods.

In all five cases, the method "Review_Finder" is not chosen because its quality-to-cost ratio is lower relative to other alternatives.

Experimental Results

We ran experiments with five different given cost

Table 1. Eight websites with different qualities and costs used in cost-based experiments.

Site	Quality	Cost
pcmall	high (3)	1.2
pczone	medium (2)	0.8
warehouse	high (3)	1.6
zdnet	low (1)	0.0
netsales	low (1)	0.6
benchin-review	high (3)	2.0
review-finder	medium (2)	1.5
cybout-review	low (1)	0.0

Table 2. BIG schedules under varying cost thresholds.

			Given Searching Cost				
Method Name	**Quality**	**Cost**	**$0**	**$1**	**$3**	**$6**	**$9**
Query_To_Zdnet	56	0	Y	Y	Y	Y	Y
Query_To_Pczone	112	0.8		Y	Y	Y	Y
Query_To_Netsales	56	0.6					Y
Query_To_Warehouse	168	1.5					Y
Query_To_Pcmall	168	1.2				Y	Y
Text_Extraction_4	40	0	Y	Y	Y	Y	Y
Text_Extraction_6	49	0	Y	Y		Y	Y
Text_Extraction_12	69	0	Y	Y		Y	Y
Text_Extraction+Cgrep_3	40	0	Y	Y	Y	Y	Y
Text_Extraction+Cgrep_4	56	0	Y	Y	Y	Y	Y
Text_Extraction+Cgrep_8	72	0	Y		Y	Y	Y
Quick_Extraction_6	22	0	Y	Y		Y	Y
Quick_Extraction_9	27	0		Y	Y	Y	Y
Quick_Extraction_19	39	0			Y	Y	Y
Benchin_Review	189	2.0			Y	Y	Y
Benchin_User_Review	175	2.0				Y	Y
Cybout_Review	41	0	Y	Y	Y	Y	Y
Review_Finder	81	1.5					
Make_Decision	90	0	Y	Y	Y	Y	Y
total cost			0	0.8	2.8	6	8.2
total quality			565	653	776	1,277	1,502

thresholds, $0, $1, $3, $6, and $9. For each specific cost, we ran the system 10 times; Table 3 presents the experiment results.

The first and second columns are the user-specified cost limit (SC) and the real searching cost (RC), respectively. The RC is very close to the SC without exceeding the limit because the scheduler was directed to put great importance on not going over the cost threshold.

The next four columns denote the number of considered products (#CP), total number of products found (#TP), aggregate information coverage (IC), and average information coverage per product object (AC). These values reflect the number of information sources used to generate the final decision. Given additional cost, BIG will adjust its searching behavior in an attempt to find both more sources of information and more supporting information for previously discovered products. The #CP and #TP did not increase when the cost increased from $1 to $3. In the $3 cases, BIG chose to spend the extra $2 on a high-quality review site to get better review information on previously discovered products, which is consistent with the semantics behind the objective criteria we supplied.

The information quality (IQ) column reflects the quality of the information sources that contributed to the decision; IQ increases according to the searching cost because high-quality sites charge a higher fee than lower quality sites. The next column denotes the extraction processing accuracy per object (PA), supplied in part by the information-processing tools. This characteristic is not affected by the searching cost, because our text-processing tools are cost-independent. Decision confidence (DC), generated by a Resun knowledge source called the decision maker, reflects the likelihood that the selected product is the best choice among the entire set of products considered. This value is based on the quality distributions of each product, and represents the chance that the expected quality is correct; it is not directly related to search cost thresholds.

The scheduling time (ST) for the first three experiments is very close to the given deadline. The last two experiments exceed the deadline because the scheduler has significantly more money to spend and can therefore expect to obtain a higher quality decision, so it chooses to spend more time. The execution time (ET) increases with cost because the increased amount of money spent makes it more likely that queried sites will contain relevant material. This in turn will increase the overall time to extract relevant information, since there will be more documents to process.

The final column shows the more frequently chosen product as the final decision in the set of 10 runs. Overall, the results indicate that the more money a client is willing to spend in gathering high-quality information, the better the final product selection decision BIG makes—although clearly this trend cannot continue forever (in the final two runs the same product was chosen).

Note also that it is quite possible to find the best product without spending any money at all. In this example we just illustrate that it is more likely that BIG will find a better product as a result of purchasing higher quality information.

Table 3. Experimental results when running BIG with different cost thresholds.

	SC	RC	#CP	#TP	IC	AC	IQ	PA	DC	ST	ET	Final decision
Average	0	0	1	3	8	1	0.33	1.83	1	667	133	Product name: Lotus FastSite 2.0
St.dev.		0	0	0	0	0	0	0	0	0	2.2	Price: 99
												Occurrence: 10/10
Average	1	0.8	8	22	28	1	0.57	1.08	0.84	600	414	Product name: Lotus Word Pro 97
St.dev.		0	0	0	0	0	0	0	0	0	35.4	Price: 63.98
												Occurrence: 6/10
Average	3	2.8	7	22	33	1.71	0.64	0.96	0.88	670	470	Product name: Lotus Word 97
St.dev.		0	0	0	0	0	0	0	0	0	11.3	Price: 63.98
												Occurrence: 8/10
Average	6	6	15.2	54	67.5	1.5	0.83	1.16	0.83	990	1041	Product name: Word Pro 97 CD WIN/W95
St.dev.		0	0.87	0	1.96	0.1	0	0.03	0	0	50.5	Price: 59.19
												Occurrence: 8/10
Average	9	8.2	21.6	69.8	80.5	1.44	0.92	1	0.85	974	1134	Product name: Word Pro 97 CD WIN/W95
St.dev.		0	1.36	0.6	0.5	0.01	0.02	0.05	0.06	0	52.1	Price: 59.19
												Occurrence: 10/10

CONCLUSIONS

The BIG system can be organized to account for complex resource-bounded constraints on its activities. In this article, we have shown the feasibility of developing complex information-gathering agents that can adjust their behavior on the basis of available processing and monetary resources. We believe that we are moving toward an information marketplace that charges for access to valuable information. As agents develop the ability to not only retrieve documents but also process their contents, the capabilities we have explored in BIG may become the norm.

The basic architecture we have laid out for resource-bounded reasoning will also be appropriate when the prices for accessing information are dynamically arrived at based on a direct negotiation with an information provider or through an auction. The DTC scheduler can be used to assess the expected effects of certain information costs on the overall decision process of an agent. This analysis is a key component in the agent deciding a reasonable price to pay for specific information and comparing alternative bids from other sites that may have different quality and coverage attributes. ■

ACKNOWLEDGMENTS

We would like to thank Norman Carver and Frank Klassner for their contributions relating to the Resun planner and the task assessor component. We would also like to acknowledge the help of Mike Chia during the formative stages of this project.

This material is based on work supported by the Dept. of Commerce, Library of Congress, and National Science Foundation under Grant No. EEC-9209623; the NSF under Grant No. IRI-9523419; and the Dept. of the Navy and Office of the Chief of Naval Research under Grant No. N00014-95-1-1198. The content does not necessarily reflect the position or policy of the Government or NSF, and no official endorsement should be inferred.

REFERENCES

1. V. Lesser et al., "BIG: A Resource-Bounded Information Gathering Agent," *Proc. 15th Nat'l Conf. on Artificial Intelligence*, AAAI/MIT Press, Cambridge, Mass., 1998, pp. 539-546; see also UMass CS Tech. Reports 98-52, 98-03, and 97-34.
2. V. Lesser et al., "BIG: A Resource-Bounded Information Gathering and Decision Support Agent," submitted for publication in *Artificial Intelligence* (special issue on Internet information agents).
3. E. Durfee et al., "Strategic Reasoning and Adaptation in an Information Economy," in *Intelligent Information Agents*, M. Klusch, ed., Springer-Verlag, 1999, pp. 176-203.
4. A. Howe and D. Dreilinger, "A Meta-Search Engine that Learns Which Engines to Query," *AI Magazine*, Vol. 19, No. 2, 1997, pp. 19-25.
5. O. Etzioni, "Moving Up the Information Food Chain: Employing Softbots on the World Wide Web," *Proc. 30th Nat'l Conf. Artificial Intelligence*, AAAI/MIT Press, Cambridge, Mass, 1996, pp. 1322-1326.
6. B. Krulwich, "The BargainFinder Agent: Comparison Price Shopping on the Internet," in *Bots and Other Internet Beasties*," J. Williams, ed., Sams.Net, 1996; also available online at http://bf.cstar.ac.com/bf/.
7. K. Decker et al., "Designing Behaviors for Information Agents," *Proc. First Int'l Conf. Autonomous Agents*, ACM Press, New York, 1997, pp. 404-413.
8. T. Wagner, A. Garvey, and V. Lesser, "Criteria-Directed Heuristic Task Scheduling," *Int'l J. of Approximate Reasoning* (special issue on scheduling), Vol. 19, No. 1-2, 1998, pp. 91-118.

Victor Lesser is a professor of computer science and director of the Multi-Agent Systems Laboratory at the University of Massachusetts at Amherst. He is a leading researcher in the areas of blackboard systems, distributed AI/multiagent systems, and real-time AI. Lesser received his PhD in computer science from Stanford University in 1972. He is a founding fellow of the American Association of Artificial Intelligence (AAAI).

Bryan Horling is a senior research fellow in the Dept. of Computer Science, University of Massachusetts at Amherst. His research interests include distributed systems and multiagent organizations. Horling received his BS in computer science and biology from Trinity College in 1996, and his MS in computer science from UMass in 1998.

Anita Raja is a PhD candidate in the Dept. of Computer Science, University of Massachusetts at Amherst. Her research interests include information gathering, resource-bounded reasoning and real-time agent control. She received her BS in computer science and mathematics from Temple University in 1996, and her MS in computer science from UMass in 1998.

Thomas Wagner is an assistant professor of computer science and a director of the Agent Institute at the University of Maine. His research interests include control technologies for intelligent software agents and information-centric agent applications. Wagner received a PhD in computer science from the University of Massachusetts at Amherst.

Xiaoqin Zhang is a PhD candidate in the Dept. of Computer Science, University of Massachusetts at Amherst. Her research interests include multiagent negotiation and coordination. She received her BS in computer science in 1995 from the University of Science and Technology of China and her MS in computer science from UMass in 1998.

Readers may contact Victor Lesser at lesser@cs.umass.edu, and Tom Wagner at wagner@umcs.maine.edu.