BIG: A Resource-Bounded Information Gathering Agent [*†]

Victor Lesser Bryan Horling Frank Klassner Anita Raja
Thomas Wagner Shelley XQ. Zhang
Computer Science Department
University of Massachusetts
Amherst, MA 01003

Abstract

Effective information gathering on the WWW is a complex task requiring planning, scheduling, text processing, and interpretation-style reasoning about extracted data to resolve inconsistencies and to refine hypotheses about the data. This paper describes the rationale, architecture, and implementation of a next generation information gathering system – a system that integrates several areas of AI research under a single research umbrella. The goal of this system is to exploit the vast number of information sources available today on the NII including a growing number of digital libraries, independent news agencies, government agencies, as well as human experts providing a variety of services. The large number of information sources and their different levels of accessibility, reliability and associated costs present a complex information gathering coordination problem. Our solution is an information gathering agent, BIG, that plans to gather information to support a decision process, reasons about the resource tradeoffs of different possible gathering approaches, extracts information from both unstructured and structured documents, and uses the extracted information to refine its search and processing activities.

Introduction

The vast amount of information available today on the World Wide Web (WWW) has great potential to improve the quality of decisions and the productivity of consumers. However, the WWW's large number of information sources and their different levels of accessibility, reliability and associated costs present human decision makers with a complex information gathering planning problem that is too difficult to solve without high-level filtering of information. In many cases, manual browsing through even a limited portion of the *relevant* information obtainable through advancing information retrieval (IR) and information extraction (IE) technologies (Callan, Croft, & Harding 1992; Larkey & Croft 1996; Cowie & Lehnert 1996; Lehnert & Sundheim 1991) is no longer effective. The time/quality/cost tradeoffs offered by the collection of information sources and the dynamic nature of the environment lead us to conclude that the user cannot (and should not) serve as the detailed controller of the information gathering (IG) process. Our solution to this problem is to integrate different AI technologies, namely scheduling, planning, text processing, and interpretation problem solving, into a single information gathering agent, BIG (resource-Bounded Information Gathering), that can take the role of the human information gatherer.

[†] This material is based upon work supported by the Department of Commerce, the Library of Congress, and the National Science Foundation under Grant No. EEC-9209623, and by the National Science Foundation under Grant No.s IRI-9523419 and IRI-9634938, and the Department of the Navy and Office of the Chief of Naval Research, under Grant No. N00014-95-1-1198. The content of the information does not necessarily reflect the position or the policy of the Government or the National Science Foundation and no official endorsement should be inferred.

Information Gathering as Interpretation

Our approach to the IG problem is based on two observations. The first observation is that a significant portion of human IG is itself an intermediate step in a much larger *decision-making process*. For example, a person preparing to buy a car may search the Web for data to assist in the decision process, e.g., find out what car models are available, crash test results, dealer invoice prices, reviews and reliability statistics. In this information search process, the human gatherer first *plans* to gather information and reasons, perhaps at a superficial level, about the time/quality/cost trade-offs of different possible gathering actions before actually gathering information. For example, the gatherer may know that Microsoft CarPoint site has detailed and varied information on the models but that it is slow, relative to the Kelley Blue Book site, which has less varied information. Accordingly, a gatherer pressed for time may choose to browse the Kelley site over CarPoint, whereas a gatherer with unconstrained resources may choose to browse-and-wait for information from the slower CarPoint site. Human gatherers also typically use information learned during the search to refine and recast the search process; perhaps while looking for data on the new Honda Accord a human gatherer would come across a positive review of the Toyota Camry and would then broaden the search to include the Camry. Thus the human-centric process is both top-down and bottom-up, structured, but also opportunistic. The final result of this semi-structured search process is a decision or a suggestion of which product to purchase, accompanied by the extracted information and raw supporting documents.

The second observation that shapes our solution is that WWW-based IG is an instance of the *interpretation problem*. Interpretation is the process of constructing high-level models (e.g. product descriptions) from low-level data (e.g. raw documents) using feature-extraction methods that can produce evidence that is incomplete (e.g. requested documents are unavailable or product prices are not found) or inconsistent (e.g. different documents provide different prices for the

same product). Coming from disparate sources of information of varying quality, these pieces of uncertain evidence must be carefully combined in a well-defined manner to provide support for the interpretation models under consideration.

In recasting IG as an interpretation problem, we face a search problem characterized by a generally combinatorially explosive state space. In the IG task, as in other interpretation problems, it is impossible to perform an exhaustive search to gather information on a particular subject, or even in many cases to determine the total number of instances (e.g. particular word processing programs) of the general subject (e.g. word processing) that is being investigated. Consequently, any solution to this IG problem needs to support reasoning about tradeoffs among resource constraints (e.g. the decision must be made in 1 hour), the quality of the selected item, and the quality of the decision process (e.g. comprehensiveness of search, effectiveness of IE methods usable within specified time limits). Because of the need to conserve time, it is important for an interpretation-based IG system to be able to save and exploit information about pertinent objects learned from earlier forays into the WWW. Additionally, we argue that an IG solution needs to support *constructive problem solving*, in which potential answers (e.g. models of products) to a user's query are incrementally built up from features extracted from raw documents and compared for consistency or suitability against other partially-completed answers – and the number of potential answers is not known *a priori.*

In connection with this incremental model-building process, an interpretation-based IG problem solution must also support sophisticated scheduling to achieve *interleaved* data-driven and expectation-driven processing. Processing for interpretation must be driven by expectations of what is reasonable, but, expectations in turn must be influenced by what is found in the data. For example, during a search to find information on word processors for Windows95, with the goal of recommending some package to purchase, an agent finding Excel in a review article that also contains Word 5.0 might conclude based on IE-derived expectations that Excel is a competitor word processor. However, scheduling of methods to resolve the uncertainties stemming from Excel's missing features would lead to additional gathering for Excel, which in turn would associate Excel with spreadsheet features and would thus change the expectations about Excel (and drop it from the search when enough of the uncertainty is resolved). Where possible, the scheduling should permit parallel invocation of IE methods or requests for WWW documents.

To illustrate our objective, consider a simple sketch of BIG in action. A simplified control flow view of this sketch is shown in Figure 1. A client is interested in finding a drawing program for Windows95. The client submits goal criteria that describe desired software characteristics and specifications for BIG's search-and-decide process. The search parameters are *quality importance = 80%, time importance = 20%, soft time deadline of 20 minutes, hard cost limitation of 0.* This translates into emphasizing quality over duration, a preference for a response in 20 minutes if possible, and a hard constraint that the search use only free information. The product parameters are: *product price: $200 or less, platform: Windows95, usefulness importance rating 100 units, future usefulness rating 25, product stability 100, value 100, ease of use 100, power features 25, enjoyability 100.* The client is a middle-weight home-office user who is primarily concerned with using the product today with a minimum of hassles but who also doesn't want to pay too much for power user features. Upon receipt of the criteria, BIG first invokes its planner to determine what information gathering activities are likely to lead to a solution path; activities include retrieving documents from known drawing program makers such as Corel and MacroMedia as well as from consumer sites containing software reviews, such as the Benchin Web site. Other activities pertain to document processing options for retrieved text; for a given document, there are a range of processing possibilities each with different costs and different advantages. For example, the heavyweight information extractor pulls data from freeformat text and fills templates and associates certainty factors with the extracted items. In contrast, the simple and inexpensive pattern matcher attempts to locate items within the text via simple grep-like behavior. These problem solving options are then considered and weighed by the task scheduler that performs quality/cost/time trade-off analysis and determines a course of action for BIG. The resulting schedule is then executed; multiple retrieval requests are issued and documents are retrieved and processed. Data extracted from documents at the MacroMedia site is integrated with data extracted from documents at the Benchin site to form a product description object for MacroMedia Freehand. However, when BIG looks for information on Adobe Illustrator at the Benchin site it also comes across products such as the Bible Illustrator for Windows, and creates product description objects for these products as well. After sufficient information is gathered, and the search resources nearly consumed, BIG then compares the different product objects and selects a product for the client. In this case, BIG's data indicates that the "best" product is MacroMedia Freehand though the academic version is the specific product that is below our client's price threshold. (The regular suggested retail price is $595.) BIG returns this recommendation to the client along with the gathered information and the corresponding extracted data.

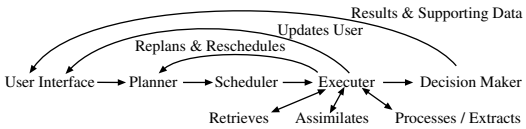

Figure 1: BIG's Problem Solving Control Flow

Though the sketch above actually illustrates one of the problem areas of BIG's text processing, that is identifying special versions of products, it illustrates one of the cornerstones of our approach to the information explosion – we believe that retrieving relevant documents is not a viable end solution to the information explosion. The next generation of information systems must use the information to make decisions and thus provide a higher-level client interface to the enormous volume of on-line information. Our work is related to other agent approaches (Wellmen, Durfee, & Birmingham 1996) that process and use gathered information, such as the WARREN (Decker *et al.* 1997) portfolio manage-

ment system or the original BargainFinder (Krulwich 1996) agent or Shopbot (Doorenbos, Etzioni, & Weld 1997), both of which work to find the best available price for a music CD. However, our research differs in its direct representation of, and reasoning about, the time/quality/cost trade-offs of alternative ways to gather information, its ambitious 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. Our time/quality/cost trade-off approach is similar to formal methods (Etzioni *et al.* 1996) for reasoning about gathering information, except that our trade-off analysis focuses on problem solving actions (including text processing) and other agent activities rather than simply focusing on the trade-offs of different information resources, i.e., our work addresses both agent control level and information value.

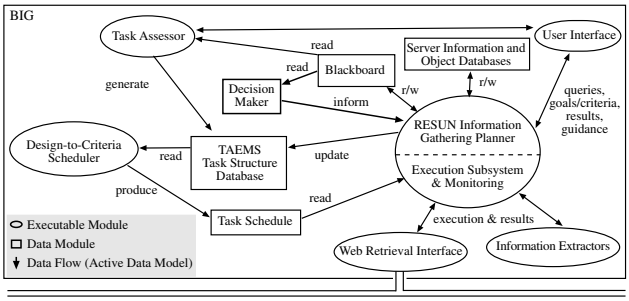

Figure 2: The BIG Agent Architecture

The BIG Agent Architecture

The overall BIG agent architecture is shown in Figure 2. The agent is comprised of several sophisticated components that are complex problem-solvers and research subjects in their own rights. The integration of such complex components is a benefit of our research agenda. By combining components in a single agent, that have hereto been used individually, we gain new insight and discover new research directions for the components. The most important components, or component groups, follow in rough order of their invocation in the BIG agent.

Task Assessor The task assessor is responsible for formulating an initial information gathering plan and then for revising the plan as new information is learned that has significant ramifications for the plan currently being executed. The task assessor is not the execution component nor is it the planner that actually determines the details of how to go about achieving information gathering goals; the task assessor is a component dedicated to managing the high-level view of the information gathering process and balancing the end-to-end top-down approach of the agent scheduler (below) and the opportunistic bottom-up RESUN planner (also below). The task assessor receives an initial information gathering goal specification from an external decision maker, which can be a human or another sophisticated automated component, and then formulates a family of plans for gathering the necessary information. The task assessor has a model of the goals that can be achieved by the RESUN planner and the performance characteristics and parameters of the actions that RESUN will employ to achieve the goals. The task assessor combines this knowledge with previously learned information stored in the server and object databases (below) and generates a set of plans that delineates alternative ways to go about gathering the information and characterizes the different possibilities statistically in three dimensions quality, cost, and duration, via discrete probability distributions. The task assessor encodes the plans in the TÆMS (Decker & Lesser 1993) generic, domain-independent task modeling framework. The TÆMS models then serve as input to the agent scheduler and other agent control components that will be added in the future (e.g., a multi-agent coordination module).

Object Database Used initially by the task assessor when determining possible courses of action, the object database is also used by the RESUN planner during information gathering sessions. As the planner creates information objects they are stored in the object database for use during future information gathering sessions. The stored objects may be incomplete and may have uncertainties attached to them, however, the uncertainties and incompletions can be filled in the next time the object is used to address a query. Through the object database and the server information database (below), BIG learns during problem solving. Information and resources learned and discovered are stored for subsequent information gathering activities. The issue of aging stored data and a detailed discussion on learning are beyond the scope of this paper.

Server Information Database The server database is used by the task assessor to help generate its initial list of information gathering options and again during the actual search process by the RESUN planner when the information gathering activities actually take place. The database is used to seed the initial search and is queried as new products are discovered. The database contains records identifying both primary and secondary information sources on the Web. Accompanying the sources are attributes that describe the sources' retrieval times and costs, their quality measures (see below), keywords relevant to the sources, and other related items. The database is constructed by an offline Web spider and modified during the search process to reflect newly discovered sites and data. This object has information aging concerns similar to those of the object database.

TÆMS Modeling Framework The TÆMS (Decker 1996) task modeling language is used to hierarchically model the information gathering process and enumerate alternative ways to accomplish the high-level gathering goals. The task structures probabilistically describe the quality, cost, and duration characteristics of each primitive action and specify both the existence and degree of any interactions between tasks and primitive methods. For instance, if the task of *Find-Competitors-for-WordPerfect* overlaps with the task of *Find-Competitors-for-MS-Word* (particular bindings of the general *Find-Competitors-for-Software-Product* task) then the relationship is described via a mutual facilitation and a degree of the facilitation specified via quality, cost, and duration probability distri-

butions. TÆMS task structures are stored in a common repository and serve as a domain independent medium of exchange for the domain-independent agent control components; in the single agent implementation of BIG, TÆMS is primarily a medium of exchange for the scheduler, below, the task assessor, and the RESUN planner.

Design-to-Criteria Scheduler Design-to-Criteria (Wagner, Garvey, & Lesser 1997; 1998) is a domain independent real-time, flexible computation (Horvitz, Cooper, & Heckerman 1989; Dean & Boddy 1988; Russell & Zilberstein 1991) approach to task scheduling. The Design-to-Criteria task scheduler reasons about quality, cost, duration and uncertainty trade-offs of different courses of action and constructs custom satisficing schedules for achieving the high-level goal(s). The scheduler provides BIG with the ability to reason about the trade-offs of different possible information gathering and processing activities, in light of the client's goal specification (e.g., time limitations), and to select a course of action that best fits the client's needs and the current problem solving context. The scheduler receives the TÆMS models generated by the task assessor as input and the generated schedule, containing parallelism where appropriate, is returned to the RESUN planner for execution.

RESUN Planner The RESUN (Carver & Lesser 1991; 1995) (pronounced "reason") blackboard based planner/problem solver directs information gathering activities. The planner receives an initial action schedule from the scheduler and then handles information gathering and processing activities. The strength of the RESUN planner is that it identifies, tracks, and plans to resolve sources-of-uncertainty (SOUs) associated with blackboard objects, which in this case correspond to gathered information and hypotheses about the information. For example, after processing a software review, the planner may pose the hypothesis that Corel Wordperfect is a Windows95 wordprocessor, but associate a SOU with that hypothesis that identifies the uncertainty associated with the extraction technique used. The planner may then decide to resolve that SOU by using a different extraction technique or finding corroborating evidence elsewhere. RESUN's control mechanism is fundamentally opportunistic – as new evidence and information is learned, RESUN may elect to work on whatever particular aspect of the information gathering problem seems most fruitful at a given time. This behavior is at odds with the end-to-end resource-addressing trade-off centric view of the scheduler, a view necessary for BIG to meet deadlines and address time and resource objectives. Currently RESUN achieves a subset of the possible goals specified by the task assessor, but selected and sequenced by the scheduler. However, this can leave little room for opportunism if the goals are very detailed, i.e., depending on the level of abstraction RESUN may not be given room to perform opportunistically at all. This is a current focus of our integration effort. In the near term we will complete a two-way interface between RESUN and the task assessor (and the scheduler) that will enable RESUN to request that the task assessor consider new information and replan the end-to-end view accordingly. Relatedly, we will support different levels of abstraction in the plans produced by the task assessor (and selected by the scheduler) so we can vary the amount of room left for RESUN's run-time opportunism and study the benefits of different degrees of opportunism within the larger view of a scheduled sequence of actions.

Web Retrieval Interface The retriever tool is the lowest level interface between the problem solving components and the Web. The retriever fills retrieval requests by either gathering the requested URL or by interacting with with both general (e.g., InfoSeek), and site specific, search engines. Through variable remapping, it provides a generic, consistent interface to these interactive services, allowing the problem solver to pose queries without knowledge of the specific server's syntax. In addition to fetching the requested URL or interacting with the specific form, the retriever also provides server response measures and preprocesses the html document, extracting other URLs possibly to be explored later by the planner.

Information Extractors The ability to process retrieved documents and extract structured data is essential both to refine search activities and to provide evidence to support BIG's decision making. For example, in the software product domain, extracting a list of features and associating them with a product and a manufacturer is critical for determining whether the product in question will work in the user's computing environment, e.g., RAM limitations, CPU speed, OS platform, etc. BIG uses several information extraction techniques to process unstructured, semi-structured, and structured information. The information extractors are implemented as knowledge sources in BIG's RESUN planner and are invoked after documents are retrieved and posted to the blackboard. The information extractors are:

- **textext-ks** This knowledge source processes unstructured text documents using the BADGER (Soderland *et al.* 1995) information extraction system to extract particular desired data. The extraction component uses a combination of learned domain-specific extraction rules, domain knowledge, and knowledge of sentence construction to identify and extract the desired information. This component is a heavy-weight NLP style extractor that processes documents thoroughly and identifies uncertainties associated with extracted data.
- **grep-ks** This featherweight KS scans a given text document looking for a keyword that will fill the slot specified by the planner. For example, if the planner needs to fill a product name slot and the document contains "WordPerfect" this KS will identify WordPerfect as the product, via a dictionary, and fill the product description slot.
- **cgrepext-ks** Given a list of keywords, a document and a product description object, this middleweight KS locates the context of the keyword (similar to paragraph analysis), does a word for word comparison with built in semantic definitions thesaurus and fills in the object accordingly.
- **tablext-ks** This specialized KS extracts tables from html documents, processes the entries, and fills product description slots with the relevant items. This KS is trained to extract tables and identify table slots for particular sites. For example, it knows how to process the product description tables found at the Benchin review site.
- **quick-ks** This fast and highly specialized KS is trained to identify and extract specific portions of regularly formatted html

files. For example, many of the review sites use standard layouts.

Decision Maker After product information objects are constructed BIG moves into the decision making phase. In the future, BIG may determine during decision making that it needs more information, perhaps to resolve a source-of-uncertainty associated with an attribute that is the determining factor in a particular decision, however, currently BIG uses the information at hand to make a decision. Space precludes full elucidation of the decision making process, however, the decision is based on a utility calculation that takes into account the user's preferences and weights assigned to particular attributes of the products and the confidence level associated with the attributes of the products in question.

Currently, all of these components are implemented, integrated, and undergoing testing. However, we have not yet fully integrated all aspects of the the RESUN planner at this time. In terms of functionality, this means that while the agent plans to gather information, analyzes quality/cost/duration trade-offs, gathers the information, uses the IE technology to break down the unstructured text, and then reasons about objects to support a decision process, it does not respond opportunistically to certain classes of events. If, during the search process, a new product is discovered, the RESUN planner may elect to expend energy on refining that product and building a more complete definition, however, it will not generate a new top down plan and will not consider allocating more resources to the general task of gathering information on products. Thus, while the bindings of products to planned tasks are dynamic, the allocations to said tasks are not. This integration issue is currently being solved. We return to this issue later in the paper.

BIG in Action

To provide a more concrete example of how BIG operates, let us walk through a sample run. The domain for this example is word processing software, where a client uses the system to find the most appropriate package, given a set of requirements and constraints. The query process begins with a user specifying search criteria, which includes such elements as the duration and cost of the search as well as desired product attributes, such as genre, price, quality and system requirements. In this example, the client desires to search for a word processor for the Macintosh costing no more than 200 dollars, and would like the search process to take about ten minutes and cost less than five dollars. The user also describes the importance of product price and quality by assigning weights to these product categories, in this case the client specified a 50/50 split between price and quality. Space precludes an in depth discussion of the product quality fields, but they include items like usefulness, future usefulness, stability, value, ease of use, power, and enjoyability.

Once these parameters are specified the query begins. The task assessor starts the process by first analyzing the user's parameters and then, using its knowledge about RESUN's problem solving options and its own top-down understanding of reasonable ways to go about performing the task, it generates a TÆMS task structure believed to be capable of achieving the query. Although not used in this example, knowledge learned in previous problem solving episodes may be utilized during this step by querying a database of previously discovered objects and incorporating this information into the task structure. The task structure produced for our example query can be seen in Figure 3. Note that sets of outcomes are associated with each method, where each outcome has a probability of occurring and is described statistically via discrete probability distributions in terms of quality, cost, and duration. This detail is omitted from the figure for clarity.

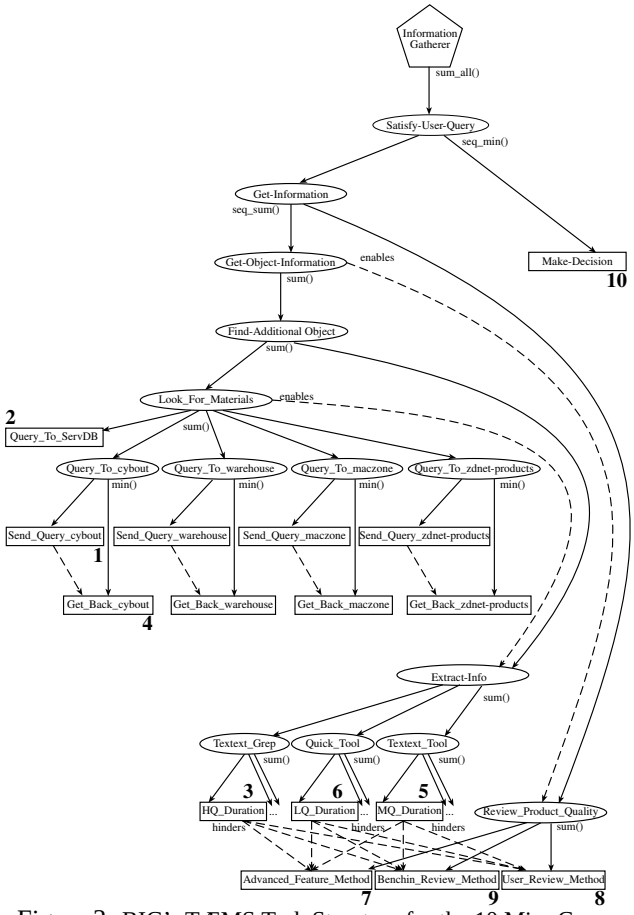

Figure 3: BIG's TÆMS Task Structure for the 10 Min. Case

Once constructed, the task structure is passed to the scheduler which makes use of the user's time and cost constraints to produce a viable run-time schedule of execution. Comparative importance rankings of the search quality, cost and duration, supplied by the client, are also used during schedule creation. The sequence of primitive actions chosen by the scheduler for this task structure is also shown in Figure 3. The numbers near particular methods indicate their assigned execution order. Again, space precludes a detailed schedule with its associated probability distributions.

The schedule is then passed to the RESUN planner/executor to begin the process of information gathering. Retrieval in this example begins by submitting a query to a known information source called "cybout." While this information is being retrieved, a second query is made and completed to the local server database information source.

This second action results in 400 document descriptions being placed on the blackboard, from which three are selected for further action. These three documents are then retrieved and processed in turn with a high-quality, high-duration sequence of information extraction tools. Before actual processing takes place, a quick search of each document's content for the product genre provides a cheap method of ensuring relevance – we envision this document preclassification step becoming more involved in the future. Three objects, one from each document, are found during the high-quality examination and placed on the blackboard. By this time, the initial query to cybout has completed and is retrieved, which results in an additional 61 documents being posted to the blackboard. Six more documents are then selected and retrieved for medium-quality, medium-duration extraction/processing. Four of these, though, fail the product genre search test and are discarded before processing takes place. Examination of the remaining two reveals two more products, which are added to the blackboard. A similar low-quality, low-duration process then adds two more objects.

At this point the system has a total of seven competing product objects on the blackboard which require more discriminating information to make accurate comparisons. To do this, three known review sites are queried for each object, each of which may produce data which is added to the object, but not combined with existing data for the given object (discrepancy resolution of extracted data is currently handled at decision time). After this, the final decision making process begins by pruning the object set of products which have insufficient information to make an accurate comparison. The data for the remaining objects is then assimilated, with discrepancies resolved by generating an average, each point being weighted by the quality of the source. A final product quality is then computed for each object, taking into account the gathered information, the quality of this information and the user's requirements. From this set the product with the highest expected quality is selected as the final recommendation. A confidence measure of this decision is also calculated based on the quality of each product and the certainty of the information. This information can be seen for several trials in Figure 4

Looking at Figure 4 in more detail one can obtain a reasonable view of how the system operates under different time constraints. In the first column of data we can see information relating to the duration of each search. Given is the user's requested duration, the duration expected by the schedule produced from the task structure and the actual execution time. Discrepancies may arise between the requested and scheduled times because of both how the task assessor creates the task structure and how the scheduler interprets it. For instance, valid 10 minute runs were available in the 600 second query, but a 743 second path was chosen because of its greater likelihood of producing high quality results. This sort of time/quality tradeoff is controlled in part by the parameters set in the user interface. The differences seen between the scheduled and actual time is caused simply by the fact that it is difficult to accurately predict the response time of remote services in the face of capricious network traffic.

The decision quality column reflects the number and qualities of the information sources used to generate the final decision. This attribute is based on the number of products considered, the number of documents used to obtain information and the quality rankings of these pages. The quality of the retrieved documents is based on knowledge about the quality of the source, which is generated by prior human examination. Unknown sites are ranked as medium quality. The product number and information coverage values increase given more scheduled time, as one would expect. The information quality values, however, may seem un-intuitive, since medium and low quality sources were used despite the fact that the quality of the information contained is known *a priori*. Such sites may be selected for retrieval for two reasons: they may respond quickly, and our set of tools may be able to analyze them particularly well. So a number of such sources may be used relatively cheaply, and still be useful when examined in conjunction with a high-quality source.

The decision confidence values describe how confident the system is in the information extraction and decision making processes. Information accuracy, supplied by the information processing tool, is the degree of belief that the actual extracted information is correctly categorized and placed in the information objects. Information confidence, generated by the decision maker, reflects the likelihood that the selected product is the optimal choice given the 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 should be noted that both these values are not dependent on the scheduled time. The accuracy does not change because our current information extraction tools do not produce different results with more execution time. Decision confidence, on the other hand, is based on the quality of the individual products, which are independent of execution time themselves, thus making the confidence independent.

The final decision of which product to recommend represents the sum of all these earlier efforts. The successes and failures of earlier processes are thus manifested here, which may lead to unpredictable results. For instance, in the five minute run, the system suggests that Adobe Acrobat will fulfill the client's word processing needs. This sort of error can be caused by the misinterpretation of an information source. Specifically, the phrase "word processing" was found associated with this package in a product description, which caused it to be accidentally included in the list of possible products. The subsequent 10 and 20 minute runs produced more useful results, both recommending the same word processor. After 40 minutes, though, the system has again selected a non-word processing package. This was also caused by a misunderstood product description, and was compounded by the fact that it was low-cost and well reviewed. It should also be noted, though, that the second and third place packages in this run were both highly rated word processors, namely ClarisWorks Office and Corel WordPerfect.

The final 5 minute query was performed after the 40 minute run, and made use of the previously generated objects when creating the initial task structure. These objects were also used to initially seed the object level of the RESUN blackboard. In this final search, more information was found on these objects, which decreased the expected qual-

Duration (seconds)			Decision Quality			Decision Confidence		
Requested	Scheduled	Actual	Num. products	Info. coverage	Info. quality	Accuracy	Info. confidence	Product retrieved
300	572	550	3	11	5 High 0 Medium 6 Low	1.461	0.830	Acrobat 3.0 Upg. from Acrobat Pro MAC CD platform: MAC price: $59.95 quality: 2.1
600	743	860	7	21	12 High 0 Medium 9 Low	1.068	0.860	Nisus Writer 5.1 Upgrade from 2.0, 3.0 or 4.0 CD ROM platform: Macintosh price: $34.95 quality: 2.7
1200	1163	942	11	25	9 High 8 Medium 8 Low	1.073	0.860	Nisus Writer 5.1 Upgrade from 2.0, 3.0 or 4.0 CD ROM platform: Macintosh price: $34.95 quality: 2.7
2400	2819	2543	23	76	28 High 16 Medium 32 Low	1.070	0.850	The Big Thesaurus V2.1 platform: Macintosh price: $27.95 quality: 2.9
Using previously learned information								
300	572	386	21	10	5 High 0 Medium 5 Low	1.058	0.710	Nisus Writer 5.1 Upgrade from 5.0 CD ROM platform: Macintosh price: $29.95 quality: 2.7

Figure 4: Five Different Results: Four with Different Time Allotments and the Fifth Generated by Using Previously Learned Knowledge

ity of the 40 minute search's erroneously selected product, The Big Thesaurus, to 2.3 from 2.9. This small amount of extra information was sufficient for the system to discount this product as a viable candidate, which resulted in a much better recommendation in a shorter period of time, i.e., the recommendation of Nisus Writer. One may also see a dramatic difference when comparing these results with the initial 5 minute query, which had similar information coverage but many fewer products to select from, which produced a lower quality decision and selected a non-word processor product.

Integration Lessons and Future Work

The integration of the different AI problem solvers in BIG, namely the RESUN planner, the Design-to-Criteria scheduler, the BADGER information extraction system, with each other and the web retriever agents, the different data storage mechanisms and process modeling systems, is a major accomplishment in its own right. The integration of these systems and tools has enabled us to study the systems in a different light than they have been studied in a stand-alone research environment. For example, the software product domain, one of BIG's IG areas, is a new domain for the BADGER extractor that required new training and new methods for handling documents, e.g., reviews and product comparisons, that are structured differently from the genres of documents dealt with in the past (e.g., terrorist articles and medical reports). We also have an interesting extraction problem when dealing with complimentary, but not competitor products. For example, when searching for word processors BIG is likely to come across supplementary dictionaries, word processor tutorials, and even document exchange programs like Adobe Acrobat. These can be misleading to the extraction tools and to BIG in general because they are referenced much like a competitor product and the documents about these products often contain terminology that further supports the notion that they are competitors rather than complimentary products. We are experimenting with enhancements to our information extraction systems to cope with this and planning to use a tf/idf style document classifier (Callan 1996) to prequalify documents before running the extraction system on them.

We have also learned new things about the Design-to-Criteria scheduler and discovered some modeling problems with applying the TÆMS task modeling framework to this application. For example, in the information gathering task structures there is a notion of search activities producing some number of documents to process, and document processing time is tied to this number of documents; additionally, the final decision making process is tied to the number of documents that are processed because with each processed document, there is some probability that it will lead to new information objects that must be considered at decision time. This dependency is data-driven and TÆMS only models certain types of domain problem solving states. We have been able to model this task adequately using existing modeling constructs, but, inaccuracies in the models sometimes lead to less-than-perfect expectations. The solution is the addition of a database resource in TÆMS that can record and model the state information pertaining to the number of documents retrieved, the number of documents processed, and the number of information objects to be considered at decision time. A secondary enhancement is the creation of new TÆMS non-local-effects to model soft task interactions, e.g., *hinders* and *facilitates*, that have an additive, rather than power-multiplier, effect.

Another major integration issue is the balance between a top-down end-to-end view of problem solving and a reactive, opportunistic view. These two views are embodied by the scheduler and the RESUN planner respectively. The scheduler designs schedules to meet real-time and real-resource performance criteria by scheduling activities from start to finish. RESUN, on the other hand, is an opportunistic problem solver that responds to newly learned information and performs processing on whatever hypothesis seems most significant at a given time step. Currently, BIG uses little of RESUN's opportunistic control to react to changes in the problem solving state. We are working on integrating the two way feedback loop between the planner, task assessor, and scheduler, that will enable the system to react, where appropriate, to changes in the problem solving state. The major issue is identifying when it is beneficial to incur the cost of rescheduling BIG's planned actions and potentially disrupting finish time guarantees that have been communicated to the client. This tension between opportunistic, bottom-up, data-driven control and top-down process-centric control is one of the major open questions in BIG but also potentially our largest gain in terms of the ability to effectively retrieve, process, and make decisions with Web-based information. Relatedly, we also intend to study a slightly different view of BIG's control as an anytime process.

As we have discussed, the integration of these components in BIG, and the view of the IG problem as an interpretation

task, has given BIG some very strong abilities. First there is the issue of information fusion. BIG does not just retrieve documents. Instead BIG retrieves information, extracts data from the information, and then combines the extracted data with data extracted from other documents to build a more complete model of the product at hand. RESUN's evidential framework enables BIG to reason about the sources of uncertainty associated with particular aspects of product object and to even work to find corroborating or negating evidence to resolve the SOUs. BIG also learns from previous problem solving episodes and reasons about resource tradeoffs. As shown, given different allotments of cost and time, and even different desired quality levels, BIG can analyze its options and plan to achieve the decision goal while meeting the client's search criteria. Though cost is not an issue spotlighted in the examples in this paper, cost on the web is a reality. For example, in the automotive product domain different sites charge different amounts for information such as invoice prices, and some sites are free, but offer less timely and less precise information.

In terms of limitations and extensibility, many of the components used in the system, such as the web retrieval interface and some of the information extractors like grep-ks and tablext-ks, are generic and domain independent. However, certain aspects of the system require domain specific knowledge and adapting BIG to operate in another domain, perhaps the auto-purchase domain, would require the addition of specific knowledge about the particular domain. For example, information extractors such as BADGER, cgrepext-ks and quickext-ks require supervised training to learn extraction rules and make use of semantic dictionaries to guarantee a certain level of performance. (Though we tested the system on the related domain of computer hardware and found it to work well considering no hardware related documents were in the training corpus.) Additionally, both the server and object databases, being persistent stores of the system's past experiences, are inherently domain dependent, rendering most of this knowledge useless and possibly distractive when used in other scenarios.

References

Callan, J. P.; Croft, W. B.; and Harding, S. M. 1992. The INQUERY retrieval system. In *Proceedings of the 3rd Intl. Conf. on Database and Expert Systems Applications*, 78–83.

Callan, J. P. 1996. Document filtering with inference networks. In *Proceedings of the 19th Intl. ACM SIGIR Conference on Research and Development in Information Retrieval*, 262–269.

Carver, N., and Lesser, V. 1991. A new framework for sensor interpretation: Planning to resolve sources of uncertainty. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, 724–731.

Carver, N., and Lesser, V. 1995. The DRESUN testbed for research in FA/C distributed situation assessment: Extensions to the model of external evidence. In *Proceedings of the 1st Intl. Conf. on Multiagent Systems*.

Cowie, J., and Lehnert, W. 1996. Information extraction. *Communications of the ACM* 39(1):80–91.

Dean, T., and Boddy, M. 1988. An analysis of time-dependent planning. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, 49–54.

Decker, K. S., and Lesser, V. R. 1993. Quantitative modeling of complex environments. *International Journal of Intelligent Systems in Accounting, Finance, and Management* 2(4):215–234.

Decker, K.; Pannu, A.; Sycara, K.; and Williamson, M. 1997. Designing behaviors for information agents. In *Proceedings of the 1st Intl. Conf. on Autonomous Agents*, 404–413.

Decker, K. S. 1996. Task environment centered simulation. In Prietula, M.; Carley, K.; and Gasser, L., eds., *Simulating Organizations: Computational Models of Institutions and Groups*. AAAI Press/MIT Press.

Doorenbos, R.; Etzioni, O.; and Weld, D. 1997. A scalable comparision-shopping agent for the world-wide-web. In *Proceedings of the 1st Intl. Conf. on Autonomous Agents*, 39–48.

Etzioni, O.; Hanks, S.; Jiang, T.; Karp, R.; Madani, O.; and Waarts, O. 1996. Optimal information gathering on the internet with time and cost constraints. In *Proceedings of the Thirty-seventh IEEE Symposium on Foundations of Computer Science (FOCS)*.

Horvitz, E.; Cooper, G.; and Heckerman, D. 1989. Reflection and action under scarce resources: Theoretical principles and empirical study. In *Proceedings of the 11th Intl. Joint Conf. on Artificial Intelligence*.

Krulwich, B. 1996. The BargainFinder Agent: Comparison price shopping on the Internet. In Williams, J., ed., *Bots and Other Internet Beasties*. SAMS.NET. http://bf.cstar.ac.com/bf/.

Larkey, L., and Croft, W. B. 1996. Combining classifiers in text categorization. In *Proceedings of the 19th Intl. Conf. on Research and Development in Information Retrieval (SIGIR '96)*, 289–297.

Lehnert, W., and Sundheim, B. 1991. A performance evaluation of text analysis technologies. *AI Magazine* 12(3):81–94.

Russell, S. J., and Zilberstein, S. 1991. Composing real-time systems. In *Proceedings of the 12th Intl. Joint Conf. on Artificial Intelligence*, 212–217.

Soderland, S.; Fisher, D.; Aseltine, J.; and Lehnert, W. 1995. Crystal: Inducing a conceptual dictionary. In *Proceedings of the 14th Intl. Joint Conf. on Artificial Intelligence*, 1314–1321.

Wagner, T.; Garvey, A.; and Lesser, V. 1997. Complex Goal Criteria and Its Application in Design-to-Criteria Scheduling. In *Proceedings of the 14th National Conf. on Artificial Intelligence*, 294–301.

Wagner, T.; Garvey, A.; and Lesser, V. 1998. Criteria-Directed Heuristic Task Scheduling. *International Journal of Approximate Reasoning, Special Issue on Scheduling*.

Wellmen, M.; Durfee, E.; and Birmingham, W. 1996. The digital library as community of information agents. *IEEE Expert*.