



## BIG: An agent for resource-bounded information gathering and decision making<sup>☆</sup>

Victor Lesser<sup>a,\*</sup>, Bryan Horling<sup>a</sup>, Frank Klassner<sup>b,1</sup>, Anita Raja<sup>a</sup>,  
Thomas Wagner<sup>c,2</sup>, Shelley XQ. Zhang<sup>a</sup>

<sup>a</sup> *Multi-Agent Systems Laboratory, Department of Computer Science, University of Massachusetts, Amherst, MA 01003, USA*

<sup>b</sup> *Department of Computing Sciences, Villanova University, 800 Lancaster Ave., Villanova, PA 19085, USA*

<sup>c</sup> *Department of Computer Science, University of Maine, Orono, ME 04469, USA*

Received 30 October 1998; received in revised form 15 June 1999

---

### Abstract

The World Wide Web has become an invaluable information resource but the explosion of available information has made Web search a time consuming and complex process. The large number of information sources and their different levels of accessibility, reliability and associated costs present a complex information gathering control problem. This paper describes the rationale, architecture, and implementation of a next generation information gathering system—a system that integrates several areas of Artificial Intelligence research under a single umbrella. Our solution to the information explosion is an information gathering agent, BIG, that plans to gather information to support a decision process, reasons about the resource trade-offs 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. © 2000 Elsevier Science B.V. All rights reserved.

*Keywords:* Information gathering; Information agents; Information systems; Planning; Scheduling; Text processing

---

<sup>☆</sup> 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. IRI-9523419 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.

\* Corresponding author. Email: lesser@cs.umass.edu.

<sup>1</sup> Email: klassner@monet.villanova.edu.

<sup>2</sup> Email: wagner@umcs.maine.edu.

## 1. Introduction and motivation

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, completeness [5], 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 [6,12,38,39] is no longer effective. The time/quality/cost trade-offs offered by the collection of information sources and the dynamic nature of the environment lead us to conclude that the user should not serve as the detailed controller of the information gathering (IG) process [41].

Our solution to the information explosion is to integrate different Artificial Intelligence (AI) technologies, namely scheduling, planning, text processing, information extraction, and interpretation problem solving, into a single information gathering agent, BIG (resource-Bounded Information Gathering) [43,44], that takes the role of the human information gatherer. In response to a query, BIG locates, retrieves, and processes information to support a decision process. To evaluate our generic approach, we have instantiated it for the software domain. BIG's area of expertise is in helping clients select software packages to purchase. For example, a client may instruct BIG to recommend a database package for Windows 98, and specify desired product attributes as well as constraints on such things as the amount of money they are willing to pay for such a product, and on the amount of time and money to spend locating information about database products. The client may also control how BIG searches by specifying a preference for information precision versus coverage. A preference for coverage will result in more products being discovered, but with less information about each product. A preference for greater precision results in BIG spending more resources to construct very accurate models of products by gathering additional corroborating information. In response to a query, BIG plans, locates, and processes relevant information, returning a recommendation to the client along with the supporting data.

The complexity of our objective mandates a high level of sophistication in the design of BIG's components. Indeed, several are complex problem solvers in their own right. A planner and associated task assessor are responsible for translating a client's information need into a set of goals and generates plans to achieve those goals. In the example above, the planner would generate plans detailing the alternative ways to fetch database product information and the alternative ways to process the information. To support reasoning about time/quality/cost trade-offs, and thus a range of different resource/solution paths, the planner enumerates several distinct plans for achieving the goals and describes them statistically in three dimensions—duration, quality, and cost—via discrete probability distributions. Another sophisticated problem-solving component, the Design-to-Criteria [61–63] (DTC) agent scheduler, examines the possible solution paths within the plan, selects a set of actions to carry out and schedules the actions—coping with an exponential scheduling problem in real-time through the use of approximation and goal directed focusing. The resulting single-agent schedule contains parallelism and

overlapping executions when the primitive actions entail non-local processing, such as requests that are issued over the network.

As BIG retrieves documents, another problem solver, an Information Extraction (IE) system [24] works in conjunction with a set of semantic, syntactic, and site-specific tools, to analyze the unstructured text documents. Information from this analysis is used for decision making and refinement of other information gathering goals.

Other complex components in BIG include a framework for modeling domain tasks, a Web server information database, and a task assessor to assist in translating the problem solver’s domain plans into a domain independent representation appropriate for use by the Design-to-Criteria scheduler and other high-level components. We will return to the agent architecture (see Fig. 3) in greater detail in Section 3.

Let us consider a high-level example to illustrate some of BIG’s capabilities and to set a context for further discussion. A client is interested in finding a word processing program for the Macintosh. The client submits goal criteria that describes desired software characteristics and specifications for BIG’s search-and-decide process. A snapshot of the system’s user specification form is given in Fig. 1.

The search parameters are: duration importance of 100%, soft time deadline of 10 minutes, hard cost limitation of \$5, and in terms of precision versus coverage, 50% of the weight is given to precision and 50% to coverage. This translates to a preference for a fast search/decision process, possibly achieved by trading off cost and quality for

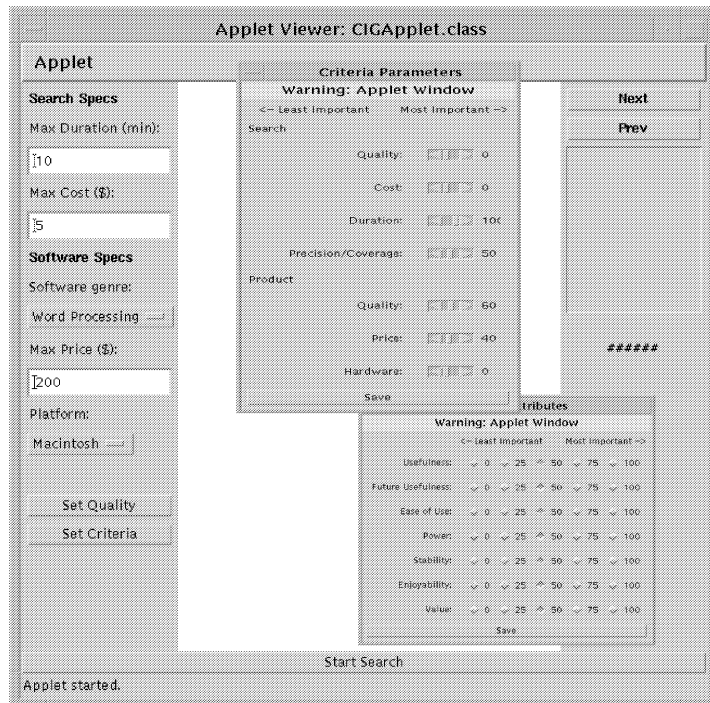


Fig. 1. BIG’s user interface.

the fast search. This also indicates that the user wants the process to take ten minutes or less and cost no more than \$5<sup>3</sup> if the search process incurs expenses when gathering information. The user also expresses no preference for coverage or precision—BIG can trade off one in favor of the other. The product parameters are: *product price \$200 or less, platform: Macintosh*. Additional product evaluation features are discussed in more detail in Section 5.5. The client is an experienced home-office user who desires a result relatively quickly and does not want to spend much money on the search, and who is primarily concerned with getting most power for the dollar in the product.

BIG's task assessor uses the supplied criteria to determine which information gathering activities are likely to lead to a solution. Candidate activities include document retrieval from known word processing software makers such as Corel and Microsoft, as well as from consumer sites containing software reviews. 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 advantages. For example, the heavyweight information extractor pulls data from free format text, fills templates and associates certainty factors from the extracted items. In contrast, the simple and inexpensive pattern matcher attempts to locate items within the text via simple grep-like behavior. BIG's task assessor handles the process of laying out these problem-solving options by emitting a task structure that describes the alternative ways to perform tasks and quantifies them statistically via discrete probability distributions in terms of quality, cost, and duration.

These problem-solving options are then considered and weighed by the scheduler—it performs a quality/cost/time trade-off analysis and determines an appropriate course of action for BIG. The schedule is executed; multiple retrieval requests are issued and documents are retrieved and processed. In this case, data extracted from documents at the Corel site is integrated with data extracted from reviews at the Benchin site to form a product description object (model) of Corel WordPerfect. Additional search and processing leads to the discovery of 14 other competing products. The decision maker, based on the product models constructed, indicates that the product that best satisfies the user's specifications is Corel WordPerfect 3.5. BIG returns this recommendation to the client along with the gathered information, corresponding extracted data, and certainty metrics about its extraction and decision processes.

The primary distinguishing characteristics of this research are:

- *Active search and discovery of information.* BIG does not rely entirely upon a pre-specified set of sites from which to gather information. BIG also utilizes general URL search engines and sites/information sources discovered during previous problem-solving sessions. Most importantly, uncertainty in the extracted information and the absence of crucial information drives further search. We provide examples in Section 4.4.
- *Resource-bounded search and analysis.* BIG problem solves to meet real-time deadlines, cost constraints, and precision, coverage and quality preferences. BIG

---

<sup>3</sup>There is no cost associated with accessing data in the experiments reported in this paper thus the cost constraint specified by the user does not alter BIG's behavior. However, as detailed in [45], experiments with BIG involving situations where accessing data from selected sites incurs cost, the cost constraints specified by the user are accounted for in BIG's information gathering process.

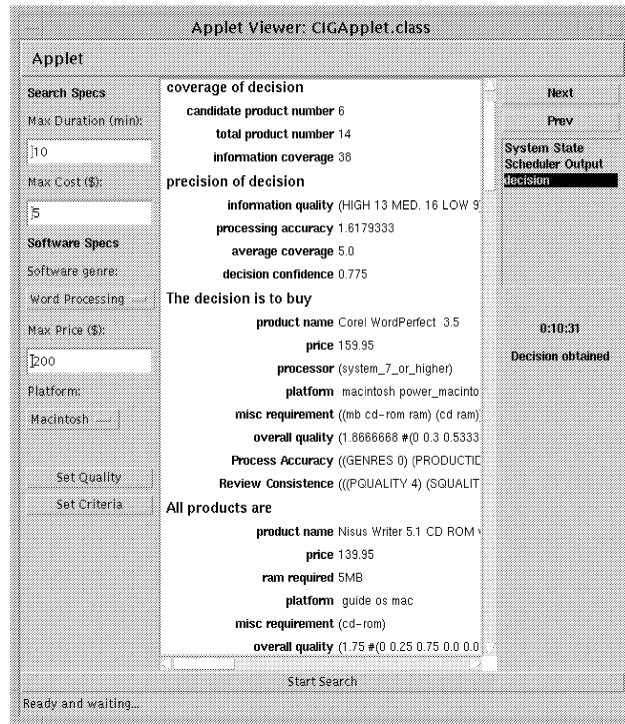


Fig. 2. BIG's final decision for sample run.

reasons about which actions to take to produce the desired result and plans accordingly. This is accomplished through the use of the Design-to-Criteria scheduler and by employing an end-to-end, rather than reactive, control process. These issues are discussed in Sections 3, 4, and 5.

- *Opportunistic and top-down control.* BIG blends opportunistic, reactive, problem-solving behaviors with the end-to-end scheduling view required to meet real-time deadlines and other performance constraints. This enables BIG to work within a high-level, structured plan without sacrificing the dynamism needed to respond to uncertainties or inconsistencies that arise in models derived from gathered information. We will discuss the details in Section 5.4.
- *Information extraction and fusion.* The ability to reason with gathered information, rather than simply displaying it for the user, is critical in the next generation of information gathering systems. BIG uses research-level extraction technology to convert free format text into structured data; the data is then incorporated and integrated into product models that are examined by BIG's decision process, resulting in a product recommendation. Details are provided in Sections 3, 4, and 5.
- *Incorporation of extracted information.* In addition to building product models, extracted information is incorporated in BIG's search as it unfolds. For example, competitor products discovered during the search are included in BIG's information

structures, possibly resulting in new goals to pursue additional information on these products. We provide further details in Sections 3 and 4, and cover an example in Section 5.

This approach to Web-based information gathering (IG) is motivated by several observations. The first observation is that a significant portion of human IG is itself an intermediate step in a much larger *decision-making process* [41]. For example, a person preparing to buy a car may search the Web to find out what car models are available, examine crash test results, obtain dealer invoice prices, or examine 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 the Microsoft CarPoint site has detailed and varied information on the relevant models, but that it is sometimes 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. A detailed discussion on the specifics of this type of opportunistic problem solving is presented in Section 5.4. Carver and Lesser in [9] provide a further exposition on the issue of exercising and balancing various types of top-down and bottom-up control.

The second observation that shapes our solution is that Web-based IG is an instance of an *interpretation problem*. Interpretation is the process of constructing high-level models from low-level data using feature-extraction methods that can produce evidence that is incomplete or inconsistent. In our current domain this corresponds to a situation where the software product descriptions generated from the raw Web documents may not contain all the desired information, or duplicate information from different sources may be contradictory. 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 Web-based 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 of the general subject that is being investigated. We first argue that an IG solution needs to support *constructive problem solving* [10,11] 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. Secondly, any solution to this IG problem needs to support reasoning about trade-offs among resource or time constraints, the quality of the selected item, and the quality of the search, analysis and decision processes. 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.

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 Windows 98, with the goal of recommending some package to purchase, an agent finding Excel in a review article that also contains Word 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, scheduling should permit parallel invocation of IE methods or requests for WWW documents.

Thus far we have outlined a large information gathering system designed to leverage the strengths of several AI subfields to address the complex task of using a large and unstructured information source like the Internet to facilitate decision making. In the remainder of this paper, we discuss related research in Section 2, and present the BIG agent architecture and its key components in Section 3. We then present a detailed execution trace of BIG in Section 4. In Section 5 we present other interesting research issues addressed by BIG using details from actual BIG runs. We also demonstrate in this section the flexibility of the architecture to different user objectives and software genres through empirical results. Conclusions and future directions are presented in Section 6.

## 2. Related research

### 2.1. Web-based information assistance

The exponential growth of the Web has not gone unnoticed by the research and commercial communities. The general solution, as one researcher so aptly put it [21], is to “move up the information food chain”, in other words, to build higher-level information-processing engines that utilize existing tools like generalized search engines (e.g., Infoseek and AltaVista). We first look at three approaches used in information-processing circles. One class of work toward this end is the *meta search engine*. Meta search engines typically issue parallel queries to multiple search engines like AltaVista and Infoseek, customizing the human client's query for each search engine and using advanced features of the search engines where available. Examples of this include SavvySearch [31] and MetaCrawler [21]; commercial meta search products are also available [33,65]. Some of these tools supplement the IR technology of the search engines—for example, if a particular advanced query technique such as phrase matching is missing from the search engine, MetaCrawler will retrieve the documents emitted from the search engine and perform its own phrase techniques on the documents. Other supplementary features provided in meta search

engines include clustering candidate documents. Since meta search engines build on the services offered by several URL search engines, their results generally offer wider Internet coverage. However, since their output is often just a list of URLs generated by the same processing techniques used in URL search engines, it tends to suffer from the same problem as the output from URL search engines themselves—too much raw data.

A second class of related work is the *personal information agent* [4,52]. Rather than making single queries to a large number of sites, these agents begin from one or more specific points on the Web and selectively pursue links in search of relevant information.<sup>4</sup> They are concept-driven, obtaining their area of interest either through hard-coded rules, explicit questionnaires or simple learning techniques. These systems are not as fast as the meta search systems, but their design goal has a somewhat different focus. Personal information agents are typically used to obtain a small number of highly relevant documents for the user to read, either all at once or continuously over an extended time period. Thus, because of the potential overhead for both link traversal and dynamic document processing these systems tend to sacrifice speed for document quality.

The third class of work addressing the information food chain is the *shopping agent*. Shopping agents typically locate and retrieve documents containing prices for specified products, extract the prices, and then report the gathered price information to the client. For example, the original BargainFinder [36] and the more recent Shopbot [20] both work to find the best available prices for music CDs. These tools often differ from meta search engines and personal information agents in that they typically do not search the Web to locate the shopping sites; instead, the systems designers develop a library containing known shopping sites and other information such as how to interact with a particular store's local search engine. Some shopping agents also integrate some of the functionality offered by the personal information agents. For example, the commercial Jango [34] shopping agent locates reviews as well as extracting price and very specific product features from vendor Web sites. Research in these systems often focuses on how to autonomously learn rules (akin to wrappers [48]) for interacting with each store's forms and for processing the output, in contrast to having a human manually encode the rules.

Consider the different attempts to move up the information food chain. The meta search engines provide information coverage, independence from the nuances of particular search engines, and speed. They also provide a measure of robustness since they are not tied to a particular search engine. Personal information agents combine IR techniques with simple heuristics to qualify documents for the client's review. Shopping agents provide information processing facilities to support the human client's information gathering objective—for example, to find the best price for a music CD. Our work extends these ideas by combining many of the characteristics that make the systems individually effective within their areas of expertise.

Like the meta search engines, BIG can use multiple different Web search tools to locate information on the Web. In contrast to the meta search engines, BIG learns about products over time and reasons about the time/quality trade-offs of different Web search options. Akin to the personal information agents, BIG gathers documents by actively searching

---

<sup>4</sup> The colloquial term “spidering” includes this directed traversal along with more undirected search strategies.



the Web (in conjunction with Web search engines). BIG, however, goes one step further by also performing information extraction on the data it retrieves. Like the shopping agents, BIG gathers information to support a decision process. However, BIG differs from shopping agents in the complexity of its decision process and in the complexity of its information processing facilities. Through IE technologies, BIG processes free format text and identifies and extracts product features like prices, disk requirements, and support policies.

## 2.2. Database research

Some aspects of BIG relate closely to issues raised by heterogeneous database systems (HDBS) [35,59]. Such databases must potentially gather data from multiple sources, which may each have different performance, content and cost. At a high level, these two problems are thus very similar. Both BIG and HDBS aim to provide transparent access to a heterogeneous set of information sources from a single access point. BIG, however, has additional concerns which HDBS typically do not address. BIG's set of information sources is more dynamic than a typical HDBS, and is composed of a mixture of search engine and single-point items. The information BIG deals with is also unstructured and noisy. As more information sources become available which are designed to be accessed by agents, HDBS techniques may become more applicable to the overall problem we are addressing.

Some of BIG's problem-solving and scheduling activities are analogous to techniques used in database query optimization. The query optimization process in a centralized database system is concerned with how best to structure information requests and manage the processing which must take place on the resulting data. In a distributed database system, a query optimizer has the additional burden of possibly choosing from among several information sources and processing locations, which each have different benefits and drawbacks [1,26,28]. These operations are analogous to the scheduling activity done in BIG, which makes similar decisions. Both tasks must consider such issues as expected server performance, data structure, activity parallelism and how best to manage the retrieved information. An important difference between a conventional query optimizer and BIG's scheduling process is the amount of user input involved. BIG uses the Design-to-Criteria agent scheduler, which takes into account the user's preferences when generating the schedule. For instance, one user may be willing to spend a lot of money in exchange for a very short but high-quality search, whereas another may be willing to spend more time to save money. DTC allows several metrics to effect its behavior, allowing a degree of customization not permitted by typical database query optimization. These trade-offs will be covered in more detail in later sections and are presented more fully in [61–63].

## 2.3. Other related issues

Technologies developed in mainstream information retrieval research may also help BIG find and extract information more reliably. Metadata information, such as RDF/PICS/XML [56] allow Web page authors to provide concise information in a format sufficiently

structured to simplify interpretation. Widespread adoption of these formats would greatly improve the effectiveness of programs like BIG. Other technologies, facilitating general inter-application (e.g., Z39.50 [2]) and inter-agent (e.g., KQML [23]) communication, can also assist by providing the standards necessary for simple information transfer. In some sense, HTTP currently fills this role, but more suitable protocols exist for the task at hand. A practical drawback with these new techniques is that they have not yet become widespread enough to make them viable. If and when standards such as RDF become widely accepted it seems clear that systems like BIG will be able to make more effective use of available information.

Grass and Zilberstein's work [25] is closely related to our basic approach, but differs in that the decision process is centered around a Bayesian network and their approach to scheduling is more reactive. BIG is also related to the WARREN [19] multi-agent portfolio management system, which retrieves and processes information from the Web. However, BIG differs in its reasoning about the 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. BIG shares some characteristics with database-centric, structured-resource approaches like TSIMMIS [27], SIMS [3], and the Information Manifold [46], but differs in that its focus is on resource-bounded information extraction and assimilation coupled with *discovery*.

The time/quality/cost trade-off aspect of our work is conceptually similar to [13,29,30,53] and formal methods [22,25] 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 concentrating only on the trade-offs of different information resources, i.e., our work addresses both agent control level and information value.

With respect to the development of digital libraries, our research is aimed at partially automating the function of a sophisticated research librarian, as in [64]. This type of librarian is often not only knowledgeable in library science but also may have a technical background relevant to the interests of the research domain. In addition to locating relevant documents for their clients, such librarians often distill the desired information from the gathered documents for their clients. They often need to make decisions based on resource concerns such as the trade-offs between billable hours and solution quality and the resource time/quality/cost constraints specified by a given client; or whether certain periodicals are available in-house, and if not, how long it will take to get them and what they will cost. We see the partial automation of a sophisticated librarian as a natural step in the evolutionary development of a fully automated digital library.

BIG also relates to research in interfaces and dialogues between human users and agents [51], though the extraction of software requirements from the user and the agent/user interaction is not the focus of this research. In the future, we envision a dynamic human/agent interface in which the client can provide online guidance to BIG to help focus the search and decision processes. In fact, BIG's architecture was designed partly to support such activities and it is one of the strengths of the flexible control paradigm used in BIG.

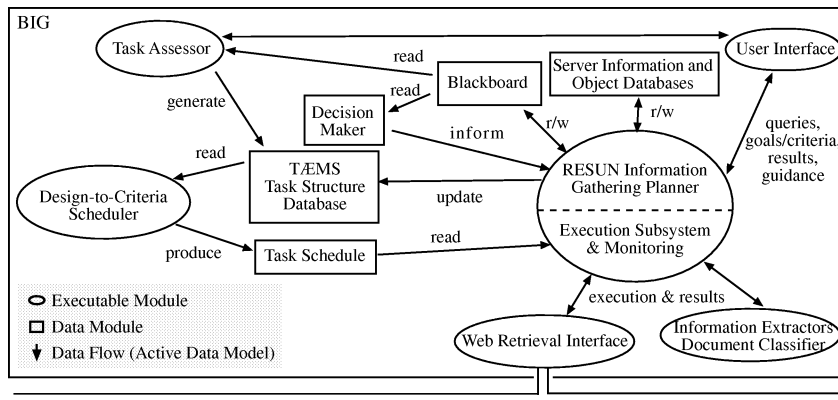


Fig. 3. The BIG agent architecture.

### 3. The BIG agent architecture

The overall BIG agent architecture is shown in Fig. 3. The agent is comprised of several sophisticated components that are complex problem solvers and research subjects in their own rights. The most important components, or component groups, follow in rough order of their invocation in the BIG agent.

**Server and object information databases.** The object database stores information objects constructed by BIG during an information gathering session. Objects represent entities from the application domain (e.g., software packages, cars) generated by BIG, about which it will make a decision. Objects may be incompletely specified; field values may be uncertain through lack of information or because of contradictory information. These uncertainties are explicitly represented as *sources of uncertainty* data structures (SOU) [7,8]. This enables BIG to plan to find information that either corroborates the current information or reduces conflicts with the current information, thereby decreasing the degree of uncertainty. The object database is also used to store information from previous searches—thus BIG can learn and improve/refine its knowledge over time.

The server information database contains numerous records identifying both primary (e.g., a review site) and secondary (e.g., URL search engine) information sources on the Internet. Within each record are stored the pertinent characteristics of a particular source, which consist of such things as its quality measures, retrieval time and cost, and relevant keywords, among others. The server database is used by the task assessor to help generate its initial sketch of information gathering options and again during the actual search process by the RESUN planner.

Both the server and object databases grow dynamically at runtime. At the start of the experimental runs described in Section 5.6, the server database is seeded with a small number (10–20) of generic information sources (e.g., vendor sites and search engines), while the object database is empty. New sources are added to the server database as they are discovered, and new characteristics about known sources (i.e., average response time, file size, references) are used to update existing entries. The object database is grown in a

similar manner by adding new products as they are found and revising records of known products. The information in these databases is part of a feedback loop, which improves the quality of data available to BIG for each query it processes. The server database is further augmented by an off-line spider process which fills the database with sources that meet general, easily checked characteristics (i.e., keyword matching, minimum textual content).

**Blackboard component.** The blackboard functions as a multileveled database for the information the system has discovered and produced thus far. Unlike the object database mentioned above, the blackboard is a runtime specific tool—it is more efficient to access, but the information will be lost when the system is shut down. Useful information from the blackboard is therefore saved into the object database for future use. Our current blackboard organization has four levels: User-Goal, Decision, Object, and Document, in order of decreasing abstraction. The layered hierarchy allows for explicit modeling of concurrent top-down and bottom-up processing, while maintaining a clear record of supporting and contradictory evidence. The information at a given level is derived from the level(s) below it, and it in turn supports the hypotheses at higher levels. For example, when evaluating the appropriateness of a particular decision hypothesis, the system examines the reliability of the text extraction processes used to generate the properties of the object. The objects themselves are each supported by the various documents from which they were generated. Fig. 4 shows the four-level structure of our current blackboard and examples of the types of objects which are stored there.

|                  |  |
|------------------|--|
| <b>User-Goal</b> | Genre: Word Processing<br>Price: \$200<br>Platform: Mac  |
| <b>Decision</b>  | Action: Buy<br>Product: Corel Wordperfect 3.5<br>Confidence: 0.775   |
| <b>Object</b>    | Product name: Corel WordPerfect 3.5<br>Overall quality: 1.87<br>Price: \$159.95<br>Platform: Macintosh<br><br><i>partial-support-sou: missing important features</i><br><i>uncertain-support-sou: uncertain about some features</i><br><i>no-support-sou: missing all information besides name</i> |
| <b>Document</b>  | URL: <a href="http://search.outpost.com/search...">http://search.outpost.com/search...</a><br>Text-content: <page text><br><br><i>no-explanation-sou: no object supported by this document</i>   |

Fig. 4. BIG's blackboard structure.

In this example, the *Corel Wordperfect 3.5* product object in the object level provides supporting evidence to the *Corel Wordperfect 3.5* recommendation made at the the decision level. There are several kinds of SOUs shown associated with the “Object” and “Document” levels in the figure; these SOUs help identify those objects which potentially require further processing. The “partial-support-sou”, for example, indicates that there are important features such as platform or processor, missing from this object. The problem solver would at some point notice this deficiency and attempt to resolve the uncertainty by retrieving and processing related documents.

**Task assessor.** The task assessor is responsible for formulating an initial information gathering plan and for revising the plan as new information is learned. The task assessor manages the high-level view of the information gathering process and balances the end-to-end, top-down constraints of the Design-to-Criteria scheduler and the opportunistic bottom-up RESUN planner (both discussed below). It heuristically generates a network of high-level plan alternatives that are reasonable, given the user’s goal specification and the desired performance objectives, in terms of time deadline and information coverage, precision and quality preferences.

The TÆMS [16] 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. TÆMS task structures are stored in a common repository and serve as a domain independent medium of exchange for the domain independent agent control component. In the single agent implementation of BIG, TÆMS is primarily used to coordinate and communicate between the scheduler (below), the task assessor, and the RESUN planner.

**Design-to-Criteria scheduler.** Design-to-Criteria [61–63] is a domain independent real-time, flexible computation [13,29,53] 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 and behavior preferences, and to select a course of action that best fits the client’s needs in the current problem-solving context. The scheduler receives the TÆMS models generated by the task assessor as input, produces a schedule in soft real-time [61], and returns the generated schedule to the RESUN planner for execution.<sup>5</sup> The resulting schedule may contain segments of parallel activities when the primitive actions entail non-local processing, e.g., issuing requests over the network. The non-local activities can be embedded within primitive actions or explicitly modeled as primitive actions with two components, one for initiation and one for polling to gather results, separated by propagation delays. This enables the agent to exploit parallelism where possible and where

---

<sup>5</sup> For a typical BIG task structure, having 25–30 primitive actions, schedule time is on the order of 10 seconds on a Digital Alphastation 6000.

the performance of the parallel activities will not adversely affect the duration estimates associated with its activities.<sup>6</sup>

In summary, the scheduler is what enables BIG to address real-time deadlines and to trade off different aspects of solution quality (e.g., precision, coverage). The scheduler does not simply trade off time and cost, it is what determines how the process should be accomplished and the appropriate time allocations given to operations or particular classes of operations (e.g., information search and retrieval versus text processing).

**RESUN planner.** The RESUN [7–9] (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 (SOU) 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 Windows 98 word processor, but associate a SOU with that hypothesis that identifies 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 ability to represent uncertainty as symbolic, explicit factors that can influence the confidence levels it maintains for hypotheses provides the cues for an opportunistic control mechanism to use in making context-sensitive decisions. For example, they might be used to adaptively engage in more unrestricted Web retrieval when a reference to a previously unknown product is encountered, or to engage in differential diagnosis to discriminate between two software products’ competitive features.

This hints at an interesting integration issue. 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 real-time [61] Design-to-Criteria 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. Improving the opportunism via a two-way interface between RESUN and the task assessor is an area of future work (Section 6). Experiments with different cost models for Web sites and scheduling with different trade-offs, using the current interface model, are presented in [45].

To work effectively, BIG must be able to perform search and discovery on the Web. The search space size and dynamism of this environment require an agent to

---

<sup>6</sup> This distinction is important because the duration estimates associated with actions are constructed assuming the dedicated efforts of the agent. In cases where multiple activities are performed in parallel, and the activities require 100% of the local processor, performance degradation will affect the actual run times of activities and result in schedules that do not perform as expected.

- (1) respond to data driven opportunities and uncertainties that arise during problem solving, and
- (2) meet real-time deadlines, address resource limitations, and trade off solution quality for time spent searching.

The RESUN planner and Design-to-Criteria scheduler combine to provide these capabilities. If the environment were static, a simple script would be sufficient to control the agent's search process.

**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 both general (e.g., InfoSeek), and site-specific search engines.

**Document classifiers.** To more effectively utilize the processing power available to it and decrease the probability of analyzing unrelated information, BIG prunes the set of documents to be processed through a series of filtering steps that impose progressively increasing processing demands and quality standards. During each stage, a test is performed, which will prevent the document from reaching the next stage if it fails. At the lowest level is a simple keyword search in the retrieved document's content. If the document fails to contain any of the supplied keywords it will fail the test. This is followed by a more sophisticated check by a Naive Bayes classifier, which is covered in detail in Section 5.1. The Naive Bayes document classifier performs statistical text classification and is provided with a set of positive and negative training documents as input. Before performing classification, the classifier indexes the data by reading the training documents and archiving a "model" containing their statistics. A document which passes these checks is then placed on BIG's blackboard. Documents selected from the blackboard will then be processed by one or more of the text extraction knowledge sources. The exact set of extractors applied to the document is governed by the document's source, or if the source is unknown to BIG, all extractors are used. This final filtering stage is responsible for the fine-grained culling of information. Pertinent details from each document are used to augment the known set of products, while the remaining content is discarded.

**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.<sup>7</sup> Documents in general are used by BIG in two different capacities: product descriptions and reviews. Different technologies optimized for use on either of these document classes are used to process the two types of documents. We determine the type

---

<sup>7</sup> The widespread adoption of XML and other structuring specifications for Web documents will help to simplify the problem of processing Web-based information.

of some documents by analyzing the site of origin. For those documents with unknown type, both review and description technologies are applied on them.

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:

- **texttext-ks.** This knowledge source processes unstructured text documents using the BADGER [55] 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. The BADGER text extractor utilizes knowledge gained from a training corpus as well as a lexicon/dictionary of domain words and their classifications in a semantic hierarchy. This component is a heavy-weight NLP-style extractor that processes documents thoroughly and identifies uncertainties associated with extracted data.  
Our main contribution in this area is how the extracted information is made useful to the rest of the system by means of back-end processing. The back-end takes the extractions made by the system and provides the degree of belief for each extraction. The degree of belief indicates the level of confidence that the extraction is accurate and is a function of the number of positive and negative training examples covered by all the rules that support a particular extraction. Using the degree of beliefs as thresholds, we determine which of the extractions are valid and also compute the certainty measure of the entire template. Also, the processed information supports opportunistic control in the sense that newly discovered information could lead to the examination of a completely different part of the solution space than before.
- **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. The cgrep knowledge source uses a lexicon/dictionary similar to that of BADGER.
- **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 built 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 constructed to identify and extract specific portions of regularly formatted html files. The quick-ks utility essentially acts as a wrapper to certain Web sites. It has knowledge about the information structure each site employs, and can efficiently extract pertinent information from these sources. The primary drawback to such a technique is the inherent difficulty in constructing such wrappers. In our system, a human expert must inspect the sites in question, deduce the structure of the pages, and then encode rules to extract the desired information. Clearly this is a labor intensive process, and one which must be repeated



for each Web site to be targeted and each time a targeted Web site alters its format. We chose to employ this technique because a relatively small number of sites could be targeted to produce a significant amount of high-quality information. Recent research in [47] has shown methods which can be employed to simplify wrapper construction and revision, which could significantly reduce the amount of effort this technology requires, thus making it more viable in a large-scale system.

**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, BIG currently uses the information at hand to make a decision. We discuss the decision process in greater detail in Section 5.5; 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. Note that we do not rigorously evaluate the final decisions that BIG produces in this paper, as we feel the issue is highly subjective. Any selected product falling within or closest to the user's desired parameters is considered a valid choice.

All of these components are implemented and integrated in BIG. The construction, adaptation, and integration of these components was a nontrivial process. BIG is a large, complex, problem-solving agent that incorporates many areas of AI research under a single umbrella.<sup>8</sup> The culmination of these efforts in BIG has produced an interesting research tool, but the integration has also influenced and refined the research directions pertaining to the individual components as well.

#### 4. Execution trace

We now describe a short sample run of the BIG system based on the high-level example described in the introduction (see Figs. 1 and 2), to better illustrate the mechanisms used both within and between BIG's components. The client is a student who uses the system to find a word processing package which will most closely satisfy a set of requirements and constraints. For clarity of presentation we describe the example trace in the following sequential stages: querying, planning, scheduling, retrieval, extraction and decision making. It is important to note that the details given below are a representative example of BIG's problem-solving techniques, and that the specific sequence of actions is highly dependent on the particular constraints and environment characteristics BIG encounters.

---

<sup>8</sup> BIG is implemented in C++, Perl, Common-Lisp, and Java. It is run on an Alphastation 6000 with 512 megabytes of RAM and requires nontrivial computing resources. However, in terms of performance, little time has been spent optimizing the system (excepting the DTC scheduler), and optimization could reduce the overhead involved with running BIG and improve BIG's ability to make better use of allocated run-time, i.e., it would be able to search more or extract more given the same resource allocation.

#### 4.1. Query formulation

Query processing is initiated when the client specifies and submits the search criteria, which includes the duration and cost of the search as well as desired product attributes such as price, quality features and system requirements. In this example the client is looking for a word processing package for a Macintosh costing no more than \$200, and would like the search process to take ten minutes and the search cost to be less than five dollars. The client also describes the importance of product price and quality by assigning weights to these product categories, in this case the client specified that relative importance of price to quality was 60% and 40% respectively. Product quality is viewed as a multi-dimensional attribute with features like usefulness, future usefulness,<sup>9</sup> stability, value, ease of use, power and enjoyability constituting the different dimensions. Such characteristics are observable through specialized analysis techniques used during the extraction phase. As seen in Fig. 1, these qualities are all equally weighted at 50 units. These are assigned relative weights of importance. The client specifies the relative importance of product coverage and precision as 20% and 80% respectively.

#### 4.2. Plan construction

Once the query is specified, the task assessor starts the process of analyzing the client specifications. Using its knowledge of RESUN's problem-solving components and its own satisficing top-down approach to achieve the top-level goal, it generates a TÆMS task structure that it finds most capable of achieving the goal given the criteria (a task structure here is akin to an integrated network of alternative process plans for achieving a particular goal). Although not used in this example, knowledge learned in previous problem-solving instances may be utilized during this step by querying the database of previously discovered objects and incorporating this information into the task structure [43].

Fig. 5 shows the TÆMS task structure produced by the task assessor in response to the client's query and the current information in the object and server information databases. The top-level task is to satisfy the user's query, and it has three subtasks: *Get-Information*, *Benchmark-Review*, and *Make-Decision*. The three subtasks represent different aspects of the information gathering and recommendation process, namely, finding information and building product models, finding reviews for the products, and evaluating the models to make a decision. The three subtasks are related to *Satisfy-User-Query* via a *seq\_sum()* quality-accumulation-function (qaf), which defines how quality obtained at the subtasks is combined at the parent task. Some qafs, like *seq\_sum()* specify the sequence in which to perform subtasks in addition to the different combinations that may be employed (the *seq* stands for "sequence"). *Seq\_sum()* specifies that all of the subtasks must be performed, in order, and that the quality of the parent task is a sum of the qualities of its children. The formal details of TÆMS are presented in [14,15], the evolving specification is at [57], and other TÆMS examples appear in [42,61,63].

---

<sup>9</sup> This relates to the openness of the software product to be compatible with newer versions of supporting software and operating system.

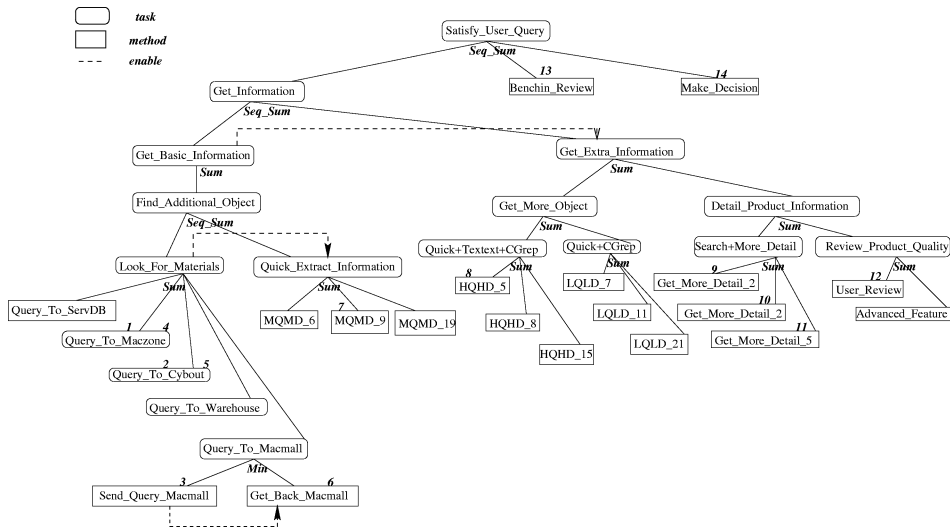


Fig. 5. BIG's TÆMS task structure for the short run.

The *Get-Information* task has two children, also governed by a *seq\_sum()*. The dotted edge leading from *Get-Basic-Information* to *Get-Extra-Information* is an *enables* non-local-effect (task interaction<sup>10</sup>) denoting that *Get-Basic-Information* must produce quality before *Get-Extra-Information* can be performed. In this case, it models the notion that product models must be constructed before any time can be spent doing optional or extra activities like improving the precision of the result or increasing the information coverage (discussed in Section 5.2). Choice in this task structure occurs any time tasks are grouped under a *sum()* qaf (there are many other qafs that entail choice, but they are not used in this example). For example, *Look-for-Materials* has six subtasks under a *sum()*, which means that any combination of these subtasks may be performed and in any order (barring deadlines on individual tasks or task interactions), i.e., the power-set minus the empty-set may be performed. Likewise with the children of *Get-More-Objects* and *Detail-Product-Information*. Alternative choices about where to search, how many places to search, which methods to employ while searching, which information extraction technologies to use, the number of reviews to gather for products, and so forth are all modeled in TÆMS. This is also what gives BIG the ability to target its performance for particular situations. For example, in a situation where a result is desired by a tight deadline, the Design-to-Criteria scheduler will analyze the task structure and find a solution path that “best” trades off quality for duration and cost. There is another element of choice in BIG: it is in the level

<sup>10</sup> The full range of task interactions expressible in TÆMS were not exploited by the task assessor component in modeling the planner's activities. One set of interactions involving facilitation/hindering we hope to use in future versions of BIG. These relationships allow us to model the fact that the degree of quality produced by a primitive task will affect in a positive/negative way the behavior of other primitive tasks. Another aspect of TÆMS that could be potentially useful in modeling IG activities is the ability to represent different outcomes associated with a task, each of which can have different types of task relationships.

of abstraction that is used in the creation of the TÆMS task structure—a *task assessor* component determines which are the options that are important to enumerate and the granularity of what is included in a leaf-node (primitive action).

#### 4.3. Schedule generation

Once generated, the task structure is then passed to the scheduler which makes use of the client'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 Fig. 5. The numbers near particular methods indicate their assigned execution order. The scheduled time and actual execution time of each method are shown in Table 1. The differences in these columns are attributable to two different sources of imprecision. The first is simply the local variance associated with Web-related activities.<sup>11</sup> The second source of imprecision is the balanced interface between the Design-to-Criteria scheduler and the opportunistic RESUN planner. To give room for RESUN to respond to data that is extracted during the search process, some of the primitive actions scheduled by DTC are not actually primitive actions. Some of the actions are instead abstractions or black boxes denoting bundles of activities. This enables RESUN to determine particular bindings as appropriate given the evolution of data during the problem-solving episode, i.e., to be data driven within the confines of the activities scheduled by DTC. One view is that DTC defines a high-level policy for RESUN that defines the major steps, and resource allocations to these, of the information gathering process. This interface is what enables BIG to respond to SOUs (sources of uncertainty) associated with extracted information and to make decisions during the search about which information to gather or which extraction processes to run—all while still staying within the time and resource guidelines set by the scheduler. Thus, at one level the schedule can be thought of as a specification of a policy that governs RESUN's activities.

#### 4.4. Information retrieval and extraction

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, MacZone ([www.zones.com](http://www.zones.com)), a computer retailer. While this information is being retrieved, a second query is made to another retailer site, Cyberian Outpost ([www.outpost.com](http://www.outpost.com)), and a third query is made to MacMall ([www.macmall.com](http://www.macmall.com)) site. Generally, queries to such sites result in a list of URLs, where each URL is accompanied by a small amount of text describing the full document. This information is combined with the query text and any other knowledge the agent has about the document such as recency, length, number of incoming links, etc., to form a *document description* object that is then put on the RESUN blackboard for consideration by other knowledge sources. The query to MacZone results in 56 document descriptions being placed on

---

<sup>11</sup> While the Web exhibits strong statistical trends during the course of a day, e.g., increasing delay time around mid-day, there may be local variance that is difficult to predict.

Table 1  
Time used for scheduling versus actual execution time in seconds

| Method name                   | Schedule time | Execution time |
|-------------------------------|---------------|----------------|
| Scheduling                    |               | 8              |
| Send_Query_maczone            | 1             | 1              |
| Send_Query_cybout             | 2             | 1              |
| Send_Query_macmall            | 1             | 0              |
| Slack_MyTime                  | 27            | 27             |
| Get_Back_maczone              | 19            | 19             |
| Get_Back_cybout               | 20            | 22             |
| Get_Back_macmall              | 19            | 8              |
| Medium_Quality_Duration_9     | 72            | 67             |
| High_Quality_Duration_5       | 51            | 49             |
| Get_More_Detail_2             | 34            | 10             |
| Get_More_Detail_2             | 35            | 58             |
| Get_More_Detail_5             | 76            | 76             |
| User-Review-Method            | 127           | 144            |
| Benchin-Review-Method         | 137           | 137            |
| Make-Decision                 | 1             | 2              |
| Total time (request time 600) | 622           | 629            |

the blackboard, the query to Cyberian Outpost results in 78 document descriptions being placed on the blackboard, while the MacMall query results in an additional 86 document descriptions being added. Out of these candidate document descriptions, 13 documents are chosen for MediumQuality(MQMD)\_9 processing. This choice is made heuristically by examining the keywords contained in the URL label and via a preference for certain Web sites (those that have yielded useful results in the past). To identify documents most likely to yield product descriptions, other heuristics, such as document recency and length could also be used.

The thirteen documents are then retrieved and run through a document classifier to determine if they are indeed word processor products; four documents are rejected by the classifier. Two of the rejected documents are translation packages, one is a description of a scanning OCR software, and the other product is a speech recognition package. These documents contain enough non-word processor related verbiage to enable the classifier to correctly reject it as a word processing product. The nine remaining (un-rejected) documents are posted on the blackboard for further consideration and processing. For example, one of these documents is: <http://search.outpost.com/search/proddesc.cfm?item=16776>; a MediumQualityMediumDuration(MQMD) text extraction process is performed on the document. The process involves using quickext-ks and cgrep-ks in sequence to create an information object that models the product. A further example of

the type of information which is stored on the blackboard can be seen in Fig. 4. After quickext-ks runs, the following object is posted:

```
Product Name : Corel WordPerfect 3.5
Price       : $159.95
DiskSpace  : 6MB
Processing Accuracy(Rating):
    PRODUCTID=0.8 PRICE=1.0 DISKSPACE=0.8
```

The cgrep-ks finds extra information about the product's processor, platform, and other miscellaneous requirements. It also finds corroborating product name and price information; this increases the processing accuracy<sup>12</sup> of these slots. After applying cgrep-ks:

```
Product Name : Corel WordPerfect 3.5
Price       : $159.95
DiskSpace  : 6MB
Processor   : -
Platform    : macintosh power_macintosh
             system_7_or_higher
misc requirement:(cd ram)
Processing Accuracy(Rating):
    PRODUCTID=1.4 PRICE=1.6 PROCESSOR=0.0
    DISKSPACE=0.8 PLATFORM=2.0 MISCREQ=1.2
```

Eight other products are found and posted on the blackboard during the execution of the MQMD\_9<sup>13</sup> method. Similarly, the method HighQualityHighDuration(HQHD)\_5 retrieves six documents, rejects one, processes five documents and posts five more products on the blackboard. At this point the system has a total of 14 competing product objects on the blackboard which require more discriminating information to make accurate comparisons. The system has, in effect, *discovered* 14 word processing products.

Those objects which are upgrades are immediately filtered out since the client did not specify an interest in product upgrades. Also, those products which are certainly not for the Macintosh platform are discarded. Subsequent efforts are focused on the remaining six products.

The three methods *Get\_More\_Detail\_1*, *Get\_More\_Detail\_2* and *Get\_More\_Detail\_5* make queries to "yahoo" and "infoseek" about the remaining products and find some review documents. A review process knowledge source is applied on every review document to extract information. The extracted information 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). For each review processed, each of the extractors generates a pair, denoted <Product Quality, Search Quality> in the information objects

<sup>12</sup> The processing accuracy values are a function of the quality of the documents and extractors used to derive the information. When obtained from a single source, the values are normalized. Concurring information from different information will result in the individual ratings being added to form the joint rating.

<sup>13</sup> There is information available about the quality and processing duration of documents to decide which documents should be selected for processing. In the current version of BIG, we did not implement this feature.

pictured below. Product Quality (PQuality) denotes the quality of the product as extracted from the review (in light of the client’s goal criteria), and Search Quality (SQuality) denotes the quality of the source producing the review. For example, if a review raves about a set of features of a given product, and the set of features is exactly what the client is interested in, the extractor will produce a very high value for the Product Quality member of the pair. Currently the Source Quality is determined based on the reference number of the document (see Section 5.5), the more widely a document is referenced, the more highly it is rated.

For example, four documents (above) are found.

- <http://www.mpp.com/mediasoft/keystone/cwp7off.htm>
- <http://www.osborne.com/whatsnew/corelwp.htm>
- <http://www.cdn-news.com/database/main/1997/2/24/0224010N.html>
- <http://www.corel.com/products/wordperfect/>

These documents are processed as review documents for product “Corel WordPerfect 3.5” and the resultant product object is:

```

Product Name : Corel WordPerfect 3.5
Price       : $159.95
DiskSpace  : 6MB
Processor   : -
Platform    : macintosh power_macintosh
              system_7_or_higher
misc requirement:(cd ram)
Processing Accuracy(Rating):
  PRODUCTID=3.8 PRICE=1.6 PROCESSOR=0.0
  DISKSPACE=0.8 PLATFORM=2.0 MISCREQ=1.8
Review Consistence:(((PQUALITY 2) (SQUALITY 3))
                    ((PQUALITY 1.2857143) (SQUALITY 2))
                    ((PQUALITY 1.2857143) (SQUALITY 2))
                    ((PQUALITY 2) (SQUALITY 2)))

```

Actually, not all of these four documents are product reviews; one of them is a list of all Corel WordPerfect products. This is caused by the weakness in general of search engines and natural language processing technologies. In this case, the only consequence of the incorrect categorization of the document is that we obtain no information after we processed it with the review extraction knowledge sources. Thus it is necessary to get information from some specific product review sites. The User-Review-Method method queries the Benchin site, producing four reviews which are processed. The document [http://www.benchin.com/\\$index.wcgi/prodrev/1112163](http://www.benchin.com/$index.wcgi/prodrev/1112163), which includes 60 users’ reviews, is selected and processed for the product “Microsoft Word 6.01”, Word 6.01 being one of the six competing products still under consideration. The new review information ((PQUALITY 2.857143) (SQUALITY 3)) is added to the “Microsoft Word 6.01” object. Similarly, Benchin-Review-Method sends queries to

the Benchin star review site (which uses a star rating system that is simple to process), producing review information for four different products.

#### 4.5. Decision making

After this phase, the final decision-making process begins by first pruning the set of product objects which have insufficient information to make accurate comparisons. The data for the remaining objects is then assimilated. Discrepancies are resolved by generating a weighted average of the attribute in question where the weighting is determined by the quality of the source. The detailed decision-making process is described in Section 5.2. The final decision is shown in Fig. 6. The decision confidence is not very high because there is a competing candidate “Nisus Writer 5.1 CD ROM with Manual” whose overall quality is only slightly less than that of “Corel WordPerfect”. This close competition degrades the decision confidence because only slight variations in search or extraction activities could have resulted in a different decision.

As will be seen in the next two sections, this information gathering process can change significantly based on the specific product specifications, the amount of time that the user

The screenshot shows the 'Applet Viewer: CIGApplet.class' window. The main content area displays the following information:

**coverage of decision**  
 candidate product number 6  
 total product number 14  
 information coverage 38

**precision of decision**  
 information quality (HIGH 13 MED. 16 LOW 9)  
 processing accuracy 1.6179333  
 average coverage 5.0  
 decision confidence 0.775

**The decision is to buy**  
 product name Corel WordPerfect 3.5  
 price 159.95  
 processor (system\_7\_or\_higher)  
 platform macintosh power\_macinto  
 misc requirement ((mb cd-rom ram) (cd ram))  
 overall quality (1.8666668 # (0 0.3 0.5333  
 Process Accuracy ((GENRES 0) (PRODUCTI  
 Review Consistence (((PQUALITY 4) (SQUALIT

**All products are**  
 product name Nisus Writer 5.1 CD ROM v  
 price 139.95  
 ram required 5MB  
 platform guide os mac  
 misc requirement (cd-rom)  
 overall quality (1.75 # (0 0.25 0.75 0.0 0.0

On the left side, under 'Search Specs', Max Duration (min) is 10 and Max Cost (\$) is 5. Under 'Software Specs', Software genre is Word Processing and Max Price (\$) is 200. The platform is Macintosh. Buttons for 'Set Quality' and 'Set Criteria' are visible. At the bottom, there is a 'Start Search' button and the status 'Ready and waiting...'. On the right side, there are 'Next' and 'Prev' buttons, a 'System State' section with 'Scheduler Output' and 'Decision' (highlighted), and a timer showing '0:10:31' with the text 'Decision obtained' below it.

Fig. 6. BIG's final decision for sample run.



is willing to have the system search, and the coverage, precision and quality criteria that are specified. Though not emphasized in the description, the amount of money the user is willing to spend accessing information sources that charge on a per-access basis can also be factored into the generation of an information gathering plan. Experiments and examples appear in [45].

## 5. Research issues in BIG's design and performance

In this section we present and discuss empirical results that demonstrate the flexibility and extensibility of the BIG approach to information gathering for decision support. Sections 5.1 and 5.7 address the issue of domain-specific knowledge and generality in BIG. Section 5.1 discusses the importance of document classification in improving system performance by filtering out inappropriate documents. Section 5.7 shows that with little additional training, new software genres can be added to BIG's library of expertise. Section 5.2 demonstrates BIG's flexibility with respect to precision and coverage. The section shows that appropriate generation of a TÆMS task structure, by the task assessor, allows the Design-to-Criteria scheduler to evaluate precision and coverage trade-offs and to meet the client objectives with respect to these. Sections 5.3 and 5.4 discuss how information fusion and opportunism manifest in BIG's information gathering activities. Section 5.5 details the process that BIG uses to evaluate client requirements and make its final product selection. Section 5.6 demonstrates empirically that the system accurately adapts its processing to address client search requirements.

It is important to note that the following sections do not attempt to evaluate system performance through comparison with an oracle, in which the best possible answer has been found by the oracle and the system is evaluated based on the proximity of its final answer to the solution returned by the oracle. Given that optimal answers are difficult to obtain in this environment, and that the overall objective is to supplement a decision process, the performance metric by which BIG is evaluated is generally whether or not the results are reasonable for a given search/query. In almost all of the situations that we have examined, BIG produces answers that are considered reasonable by a human decision maker for the given search and product criteria.

### 5.1. The importance of document classification

Until recently, BIG has been plagued by an interesting extraction problem when dealing with products that are complimentary to the class of products in which a client is interested. 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 products are misleading because their product descriptions and reviews often contain terminology that is very similar to the terminology used to describe members of the target class. When BIG processes one of these misleading documents, it gets *distracted* and future processing is wasted in an attempt to find more information about a product that is not even a member of the target class. For example, if BIG encounters a reference to Adobe Acrobat when searching for word processors,

|                         | # Rejected | Top Candidates   | Selected Product             |
|-------------------------|------------|--|------------------------------|
| No Filter or Classifier | 0/13       | Portuguese Dictionary Module<br>Norwegian (Nynorsk) Dictionary Module<br>Norwegian (Bokmal) Dictionary Module<br>The Nisus Dictionary Collection<br>US Definition Dictionary | Portuguese Dictionary Module |
| Simple Filter           | 31/44      | EndLink 2.0<br>Spelling Coach Pro 4.1<br>Retrieve It! 2.5<br>Nisus Writer 5.1 CD ROM with Manual<br>Microsoft Word 6.01  | EndLink 2.0                  |
| Filter & Classifier     | 61/74      | ClarisWorks Office 5.0<br>Corel WordPerfect 3.5 ACADEMIC<br>Nisus Writer 5.1 CD ROM with Manual<br>Corel WordPerfect 3.5   | ClarisWorks Office 5.0       |

Fig. 7. Advantages of document classification.

and then elects to retrieve the product description for Acrobat, the extraction techniques are likely to yield data that seems to describe a word processor. Subsequently, BIG may elect to gather more information on Acrobat, further degrading the overall efficiency of the system. Experiments indicate that this type of distraction can be reduced through the use of a document classifier before text extraction is performed on candidate documents. Documents that do not seem to be members of the target class are rejected and text extraction is not performed on them—thus no new distracting information objects are added to BIG’s blackboard.

Fig. 7 provides a sample of our initial results. BIG was run in three different modes:

- (1) BIG alone,
- (2) BIG with the use of a simple grep-like pattern-matching filter to classify documents,
- (3) BIG with the use of Naive Bayes document classifier [49] and the simple grep filter.

The grep-like filter examines the document for instances of terms that describe the software genre in question, e.g., “word processor”. These terms are hand produced for each query genre—in essence, hardwired into the system. In contrast, the document classifier is trained using positive and negative examples—it learns term-based similarity and difference measures. In all three modes, BIG has decided that it has time to process 13 documents in total for the given search parameters. When filtering and classification of documents results in certain documents being rejected (rows two and three in Fig. 7), a larger corpus of documents is examined (44 and 74 respectively) to obtain the target number of documents.

In the first run, shown in the figure, neither filter nor classifier are used. All documents retrieved are processed by the information extractors. None of the top five objects in this test case are members of the target product class—they are all related to word processors but none of them is actually a word processing product. Clearly, BIG does very poorly when relying on outside sources like vendor’s search engines to classify products. In the second run, the simple grep-like filter is used to check documents before processing; 31 documents are rejected by the filter and the overall results are a little better. There are word processing products among the candidates, but the selected product is not a word processor. In the last run, both classifier and filter are used to check documents; 53 documents are

rejected. All of the top-ranked candidates are word processing products and the top product, “ClarisWorks Office 5.0”, is an integrated office suite that includes a word processing package.

Clearly, document pre-classification is necessary to filter retrieved documents before they are used to produce product objects. Vendor search engines are typically keyword based and are therefore prone to return numerous products that are not members of the target class but are instead related or supplementary products. Improving the classification of documents and widening the training corpus for the classifier are areas of future development.

The classifier can be applied to other domains, however, it requires a new training corpus. This is true with other text processing knowledge sources and document classifiers as well—components based on statistical properties of text (akin to IR tf/idf statistics) require training corpora in order to apply them to a different domain. While such training requires hands-on person hours, it is reasonable to assume that a library of such classifications for new domains could be compiled over time, allowing the capabilities to grow as needed.

To explore this issue, we added new genres to BIG’s library of expertise using a simple procedure. A query for the new genre, e.g., image-editing software, is given to BIG. BIG then gathers information on image-editing software by submitting various specified keyword queries to general search engines and by looking at software makers and review sites. Of course, when BIG retrieves the documents, they are filtered out by BIG’s existing set of document classifiers. However, this process yields a large pool of documents that can then be classified by hand and used to train the document classifiers on the new genre. Using this process, it is possible to integrate a new software genre in a little less than an hour’s time. The text extraction tools are generic enough to handle the new genre and no new training documents or additions to the lexicon were required. Currently, no special tools are being used to automate this process of integration. The performance of the system on the new genre is described in the experimental results in Section 5.7. As part of our future work, we foresee developing mechanisms to allow users to provide feedback about the correctness of the decision process and which products selected by the system are in the ball-park for a new genre. This information can be used incrementally as we get new/more users who access this genre.

## 5.2. *Precision versus coverage*

*Precision versus coverage* is an issue often discussed in literature relating to information gathering or information retrieval. In the BIG context, once a satisfactory amount of information has been processed to support a high-quality decision process, the issue becomes how best to spend the remaining time, cost, or other constrained resources. One alternative is to spend the time gathering more information about other products, i.e., discovering new products and building models of them. Another alternative is to spend the time discovering new information about the existing products in order to increase the precision of the product models. Both alternatives can lead to higher-quality decision processes since both expand the range of information on which the decision is based.

BIG supports both of these behaviors, and a range of behaviors in between the binary extremes of 100% emphasis on precision and 100% emphasis on coverage. BIG clients specify a precision/coverage preference via a percentage value that defines the amount of “unused” (if there is any) time that should be spent improving product precision. The remainder is spent trying to discover and construct new products. For example, if a client specifies 0.3, this expresses the idea that 30% of any additional time should be spent improving precision and 70% should be spent discovering new products.

BIG achieves this trade-off behavior in two ways: by planning and scheduling for it *a priori*, and by responding opportunistically to the problem-solving context within the constraints of the schedule. Scheduling for the precision/coverage trade-off is accomplished by relating the precision and coverage specification to quality<sup>14</sup> for the Design-to-Criteria scheduler and giving the scheduler a set of options, from which to choose a course of action. In Fig. 5, *Get-Extra-Information* has two subtasks, *Get-More-Objects* and *Detail-Product-Information* denoting the two different ends of the spectrum. *Get-More-Objects* represents the coverage end and *Detail-Product-Information* represents the precision end. The *sum()* quality accumulation function under the parent task, *Get-Extra*, models that the scheduler may choose from either side depending on the quality, cost, duration, and certainty, characteristics of the primitive actions under each. Client precision/coverage preference is related to quality for the primitive actions under these tasks, e.g., the actions pertaining to precision receive higher quality when increased weight is given to precision. This approach enables the scheduler to reason about these extra activities, and their value, and relate them to the other problem-solving options from a unified perspective. Thus, the overall value of pre-allocating “extra” time to coverage or precision is also considered in light of the other candidate activities.

BIG can also work opportunistically to improve coverage or precision, as is described in Section 5.4. A third option, not currently implemented, is for BIG to revise its problem-solving options and reschedule as new information is gained and the context (state of the blackboard, environment, time remaining, etc.) changes. This would enable BIG to react opportunistically but to do so wholly in the context of reasoning about the quality, cost, duration, and certainty trade-offs of its options from a unified perspective.

Table 2 shows BIG’s ability to trade off precision and coverage. In providing this data, we are not attempting to generalize in this section that any particular trade-off between the two is better than the other, only that such a trade-off exists. We feel this characteristic is interesting both because of the way BIG implements and exhibits the behavior, and because of the ramifications it has on how users can control a search process. The table contains data for three sets of runs, for the same query and with the same criteria settings (only the precision setting is varied). In each run, three trials are performed, each with a different precision preference setting, namely 10%, 50%, and 90% respectively. Since network performance varies during execution, and there is some element of stochastic behavior in BIG’s selection of equally ranked documents, no two trials are identical even if they

---

<sup>14</sup> The particular values associated with the qualities of primitive actions is not critical provided that the relative relationships among qualities of different actions are consistent with the domain. The purpose of the quality attributes and qafs are to give the scheduler a sound basis for making trade-offs among quality, cost and time characteristics of different schedules.

Table 2  
Trading off precision and coverage

| # | DRatio | Scheduled | Execution | T.P. | #P. | A.C. | P.A. | D.C. |
|---|--------|-----------|-----------|------|-----|------|------|------|
| 1 | 0.1    | 629       | 587       | 33   | 7   | 1.86 | 1.38 | 0.85 |
|   | 0.5    | 622       | 720       | 14   | 6   | 3.83 | 1.47 | 0.89 |
|   | 0.9    | 651       | 685       | 8    | 3   | 7.0  | 2.12 | 0.89 |
| 2 | 0.1    | 629       | 656       | 33   | 8   | 1.75 | 1.32 | 0.85 |
|   | 0.5    | 622       | 686       | 14   | 4   | 3.0  | 1.5  | 1    |
|   | 0.9    | 652       | 522       | 7    | 1   | 7.0  | 2.12 | 1    |
| 3 | 0.1    | 629       | 702       | 29   | 7   | 1.71 | 1.47 | 0.85 |
|   | 0.5    | 622       | 606       | 15   | 6   | 2.33 | 1.52 | 1    |
|   | 0.9    | 651       | 572       | 7    | 2   | 4.5  | 1.7  | 0.99 |

Key: # is the run number, DRatio = preference for precision, Scheduled = total execution time as predicted by model and anticipated by scheduler, Execution = actual execution time, T.P. = total product objects constructed, #P = total products passed to decision process, A.C. = average coverage per object, P.A. = extraction processing accuracy per object, D.C. = overall decision process confidence.

have the same preference settings. Note the general trends in the different runs. As more weight is given to increasing precision, the number of products (T.P.) decreases, as does the number of products used in the decision process (#P). The difference between these two values is that some product objects lack sufficient information to be included in the decision process and some of the product objects turn out to relate to products that do not meet the client's specification (e.g., wrong hardware platform, wrong product genre, price too high, etc.). An extreme example of this is in run number two in the third trial, where only one product is produced. As the number of products decrease as more weight is given to precision, the average information coverage per object (A.C.) *increases*, as does the information extraction/processing accuracy (P.A.). The decision confidence also generally increases, particularly in runs two and three, though this item takes into account the total coverage represented by the products as well as the precision of the product models so its increase is not proportional to the other increases.

Schedules for the 10% and 90% precision runs (respectively) are shown in Figs. 8 and 9. The schedules show the sequence of primitive actions and their start times (as expected values rather than distributions). The schedules diverge on or around time 36 where schedule 8 begins a series of *Medium\_Quality\_Duration* and *Low\_Quality\_Duration* activities that retrieve and process additional product related documents. The postfixed integers on the method names (e.g., *method10\_Medium\_Quality\_Duration\_6*) denote the number of documents (e.g., 6) that will be retrieved by the method. This series of steps results in the production of nearly twenty additional product description objects. In contrast, around that same time, schedule 9 begins a series of *Get\_More\_Detail* actions that seek to find information about existing product objects.

| Start Time<br>(expected value) | Method Name                        |
|--------------------------------|------------------------------------|
| 0                              | method6_Send_Query_maczone         |
| 1                              | method4_Send_Query_cybout-product  |
| 3                              | method2_Send_Query_warehouse       |
| 4                              | method0_Send_Query_macmall         |
| 6                              | Idle (awaiting results)            |
| 31                             | method7_Get_Back_maczone           |
| 32                             | Idle (awaiting results)            |
| 33                             | method5_Get_Back_cybout-product    |
| 35                             | method3_Get_Back_warehouse         |
| 36                             | method1_Get_Back_macmall           |
| 37                             | method10_Medium_Quality_Duration_6 |
| 93                             | method9_Medium_Quality_Duration_8  |
| 165                            | method21_Medium_Quality_Duration_6 |
| 221                            | method20_Medium_Quality_Duration_8 |
| 296                            | method18_Low_Quality_Duration_7    |
| 346                            | method15_Get_More_Detail_2         |
| 386                            | method14_Get_More_Detail_2         |
| 429                            | method22_Benchin-Review-Method     |
| 631                            | method23_Make-Decision             |

Fig. 8. Schedule for 10%/90% precision to coverage.

| Start Time<br>(expected value) | Method Name                        |
|--------------------------------|------------------------------------|
| 0                              | method6_Send_Query_maczone         |
| 1                              | method4_Send_Query_cybout-product  |
| 3                              | method2_Send_Query_warehouse       |
| 4                              | method0_Send_Query_macmall         |
| 6                              | Idle (awaiting results)            |
| 31                             | method7_Get_Back_maczone           |
| 32                             | Idle (awaiting results)            |
| 33                             | method5_Get_Back_cybout-product    |
| 35                             | method3_Get_Back_warehouse         |
| 36                             | method1_Get_Back_macmall           |
| 37                             | method9_Medium_Quality_Duration_8  |
| 111                            | method21_Medium_Quality_Duration_6 |
| 165                            | method15_Get_More_Detail_2         |
| 206                            | method14_Get_More_Detail_2         |
| 247                            | method13_Get_More_Detail_5         |
| 347                            | method11_User-Review-Method        |
| 508                            | method22_Benchin-Review-Method     |
| 646                            | method23_Make-Decision             |

Fig. 9. Schedule for 90%/10% precision to coverage.

From an end user perspective, the precision/coverage specification enables clients to express preferences for one solution class over another. For a client who needs a speedy result, and has an accordingly short deadline, the preference specification may result in

---

```

URL_A http://www.cc-inc.com/sales/detail.asp?dpno=79857&catalog_id=2
URL_B http://www.freedombuilders.com/dramatica.htm
URL_C http://st2.yahoo.com/screenplay/dpro30mac.html
URL_D http://www.heartcorps.com/dramatica/questions_and_answers/dramatical0.htm
URL_E http://www.zdnet.com/macuser/mu_0796/reviews/review12.html
URL_F http://www.macaddict.com/issues/0797/rev.dramaticapro.html

```

---

Fig. 10. URLs for documents retrieved during processing.

a slight difference at best. However, for a client with more generous time resources, the difference can be pronounced.

### 5.3. Information fusion

We use the term *information fusion* to denote the process of integrating information from different sources into a single product object; the information may be complimentary, but also contradictory or incomplete. There are several aspects to the fusion issue. The most straightforward type of fusion is information addition—where a document provides the value to a slot that is not yet filled. A more interesting type of fusion is dealing with contradictory single value information, e.g., two documents reporting different prices for a product, or two documents identifying a different production company for the product. When BIG encounters this fusion issue, the item with the highest associated degree of belief is used.<sup>15</sup> Another issue is how to integrate different opinions about the product. The latter is done in BIG by associating two metrics with each review document, one representing information or site quality, and one representing the quality of the product as expressed in the review. This dual then conceptually represents a value/density pair—the information quality metric determines the weight given to the product quality metric when comparing different metrics for different reviews. To illustrate BIG’s fusion process, consider the following partial trace.

In this example, BIG is searching for word processor products for the Macintosh. In response to a general query about word processing products, the *MacMall* retail site returns a list of URLs. URL\_A, from Fig. 10, is selected by BIG for retrieval and processed. BIG extracts “Dramatica Pro 2.0” from the document as the title of the software package; it also extracts that “Screenplay” (Inc.) is the maker and that the package sells for a price of \$289.99.<sup>16</sup> The result of this extraction is the partial product object shown in Fig. 11(a).

The values in the *Processing Accuracy* slots are certainty factors denoting the quality and certainty of the extraction process that filled the respective slots. Since the document provides very little additional information about Dramatica, BIG associates an *uncertain-support* SOU with the object. Because the product object is a promising area of exploration, relative to everything else on the blackboard, BIG decides to attempt to resolve the SOU. Toward that end, it queries Infoseek about Dramatica, resulting in a long

<sup>15</sup> In the future, we hope to explore the use of RESUN’s opportunistic control to handle this situation by trying to retrieve additional information to resolve the conflict.

<sup>16</sup> Dramatica is actually a product contained in our corpus of word processor class documents used to train the document classifier. Thus, the pursuit of Dramatica as a word processing package is valid from BIG’s perspective, though the classification of Dramatica as a word processor is perhaps debatable.

---

```

Product name: DRAMATICA PRO 2.0 3.5IN DSK
Company name: Screenplay
Price: 289.99
Processing Accuracy: (GENRES 0) (PRODUCTID 0.8) (COMPANYID 1.0) (PRICE 2.2)
                    (PRODUCTDESC 0) (PROCESSOR 0) (RAMREQ 0) (DISKSPACE 0)
                    (PLATFORM 0) (MISCREQ 0) (OVERALLQUAL 0)

```

---

## (a) Initial product object

---

```

Product name: DRAMATICA PRO 2.0 3.5IN DSK
Company name: Screenplay
Price: 289.99
Processor:
Platform: macintosh-system_7.0_or_higher windows_95
Misc requirement: (mb ram)
Overall quality: -0.5714286
Usefulness: 0
Future Usefulness: -1
Ease of Use: -1
Power: -1
Stability: -1
Enjoy ability: 0
Value: 0
Process Accuracy: (GENRES 0) (PRODUCTID 0.8) (COMPANYID 1.0) (PRICE 2.2)
                  (PRODUCTDESC 0) (PROCESSOR 0) (RAMREQ 0) (DISKSPACE 0)
                  (PLATFORM 0) (MISCREQ 0) (OVERALLQUAL 0)
Review Consistence: ((PQUALITY -0.5714286) (SQUALITY 1))

```

---

## (b) Product object after two documents

---

```

Product name: DRAMATICA PRO 2.0 3.5IN DSK
Company name: Screenplay
Price: 289.99
Processor:
Platform: macintosh-system_7.0_or_higher windows_95
Misc requirement: (mb ram)
Overall quality: 1.4285715
Usefulness: 1
Future Usefulness: 1
Ease of Use: 1
Power: 2
Stability: 2
Enjoy ability: 1
Value: 1
Process Accuracy: (GENRES 0) (PRODUCTID 0.8) (COMPANYID 1.0) (PRICE 2.2)
                  (PRODUCTDESC 0) (PROCESSOR 0) (RAMREQ 0) (DISKSPACE 0)
                  (PLATFORM 1.8) (MISCREQ 1.2) (OVERALLQUAL 0)
Review Consistence: ((PQUALITY 1.4285715) (SQUALITY 1))
                  ((PQUALITY 2) (SQUALITY 2))
                  ((PQUALITY -0.5714286) (SQUALITY 1))

```

---

## (c) Intermediate product object

---

```

Product name: DRAMATICA PRO 2.0 3.5IN DSK
Company name: Screenplay
Price: 289.99
Processor:
Platform: macintosh-system_7.0_or_higher windows_95
Misc requirement: (mb ram)
Overall quality: 2.857143
Usefulness: 3
Future Usefulness: 2
Ease of Use: 5
Power: 2
Stability: 0
Enjoy ability: 4
Value: 4
Process Accuracy: (GENRES 0) (PRODUCTID 0.8) (COMPANYID 1.0) (PRICE 2.2)
                  (PRODUCTDESC 0) (PROCESSOR 0) (RAMREQ 0) (DISKSPACE 0)
                  (PLATFORM 1.8) (MISCREQ 1.2) (OVERALLQUAL 0)
Review Consistence: ((PQUALITY 2.857143) (SQUALITY 3))
                  ((PQUALITY 0.71428573) (SQUALITY 3))
                  ((PQUALITY 1.4285715) (SQUALITY 1))
                  ((PQUALITY 2) (SQUALITY 2))
                  ((PQUALITY -0.5714286) (SQUALITY 1))

```

---

## (d) Final product object

Fig. 11. Evolution of the Dramatica product object.



list of URLs that are combined with their descriptive text to create candidate document description objects which are added to the blackboard. BIG selects and retrieves a subset of these, starting with URL\_B, which is a detailed description of the product. Processing the description results in the addition of platform specifications to the product object, namely that it runs on Windows 95 and Apple Macintosh systems. The description also contains sufficient verbiage that it is analyzed using a keyword-based review processing heuristic that looks for positive and negative phrases and rates products accordingly, weighing the product features by the user preference for such features. Though the verbiage praises the product, it is given a rating of  $-0.57$  because the review does not praise the product for the features in which the client is interested. In other words, even though the review is positive, it does not make specific reference to the product features in which the client is interested—such as a specific platform or program characteristic—and thus it is given a negative value to denote that the product is below average quality-wise. However, since the document in question is not widely referenced by other documents, it is given a low information quality (source quality) rating and the negative review (product quality) rating will thus have little weight when compared to other sources. The product object after this step is shown in Fig. 11(b).

In response to the continued existence of the *uncertain-support* SOU, BIG decides to gather more information. It selects and retrieves URL\_C, URL\_D, URL\_E, and URL\_F, in that sequence. Space precludes presenting an exhaustive sequence of product object transformations as information is integrated into the object. Fig. 11(c) is the result after processing the review at URL\_D. Note the elevation of the product's overall quality rating and the increase in the various rating criteria like ease-of-use and stability. For free format reviews such as this one (in contrast to sites that employ consistent numerical rating systems), these metrics are determined by a set of heuristics that examine the text for certain positive or negative expressions.

The remaining documents are retrieved, processed, and integrated in a similar fashion. The product object after processing all of the selected documents is shown in Fig. 11(d). For example, the final product object is subsequently compared to other product objects during the decision process (see Section 5.5). While this example results in the construction of a fairly complete product object, the objects used in the final decision process are not all at the same level of completeness. Some objects may contain less information (but not much) and some may contain more product details or more review summarization statistics. The decision process takes into account the quantity and quality of the information pertaining to the objects.

#### 5.4. Opportunism

As discussed, opportunism in the BIG system currently occurs within the boundaries of the initial schedule. The primitive actions seen by the scheduler are often abstractions of sets of operations that BIG plans to perform, thus enabling BIG to respond opportunistically during the execution of these actions to newly gathered data or changes in the environment. To illustrate, consider a simple example where BIG is gathering documents to

recommend a word processor. A portion of the schedule (without the numerical detail), produced to address the specified resource constraints, follows:

```
-----
... | Get_Back_macmall | MQMD_method|Get_More_Detail_1 |
    | Benchin-Review-Method | Make-Decision | ...
-----
```

As a consequence of executing the schedule, documents are retrieved from the *MacMall* site and processed using medium quality, medium duration text extraction techniques (meaning a set of simple and more sophisticated extractors), denoted by the *MQMD\_method* in the schedule. The product name, “Nisus Writer 5.1 CD ROM with Manual” is extracted from one of the documents and is posted as an object on the blackboard. Since the product name is the only information that could be extracted from the document at hand, a *no-support* SOU is attached to the object, signifying the need to obtain more detailed information on the product in order for it to be used in the final decision process.

As BIG actively pursues and plans to resolve SOUs, method *Get\_More\_Detail\_1* is selected for execution to resolve the SOU. The method looks for objects which contain the *no-support* SOU and tries to find more information on related products by retrieving and extracting documents. In this particular example, *Get\_More\_Detail\_1* queries InfoSeek with the keywords “Nisus Writer”, resulting in the production of a set of candidate URLs and partial document descriptions. BIG decides to retrieve and process the review located at URL [32]. Text processing of this document leads to the discovery of two new potential competing products, namely “Mac\_Publishing” and “WordPerfect” thus two more objects with the product name slots filled are posted to the blackboard accompanied by *no-support* SOUs as the product objects are essentially empty at this time.

BIG now has the following options:

- (1) It can continue with its original schedule, which entails executing the *Benchin\_Review\_Method* to gather reviews for the Nisus Writer product, or,
- (2) it can make an opportunistic change in its plans and find more information on objects which contain unresolved *no-support* SOUs.

In this case, this would mean executing the *Get\_More\_Detail\_1* method for the *Mac\_Publishing* and *WordPerfect* objects. The choice is determined by the precision versus coverage specification (Section 5.2) as well as how much time is available to perform this extra processing before the deadline.

In this particular scenario, the latter choice is made and BIG decides to find more information on the new products rather than follow the original schedule. Method *Get\_More\_Detail\_1* is executed and the Cyberian Outpost retail site is queried for information on the two products. The query for “Mac\_Publishing” returns no supporting information and the certainty that it is a valid word processing product is decreased. The query for “WordPerfect”, on the other hand is supported by the document <http://srch.outpost.com/search/proddesc.cfm?item=30271> and thus the belief that the product is a word processing product is unchanged. Processing of the document produces new information about the product, shown in Fig. 12.

The information is incorporated into the product object and BIG continues processing its initially scheduled activities. However, BIG may later elect to work on the WordPerfect product object again as it is now a valid candidate product.

|           |   |
|-----------|---|
| PRODUCTID | Corel WordPerfect 3.5 - ACADEMIC  |
| PRICE     | 29.95   |
| MISCREQ   | UNINITIALIZED   |
| SUPPORT   | 1   |
| SOURCE    | <a href="http://srch.outpost.com/search/proddesc.cfm?item=30271">http://srch.outpost.com/search/proddesc.cfm?item=30271</a> |

Fig. 12. Information produced.

### 5.5. BIG's decision process

The decision maker knowledge source decides which product should be recommended to the user after the search process is completed. Generally, the decision maker looks at each product and calculates a total score representing the overall level of consistency with the client's query. As there are several features for one product, such as price, quality, and hardware (platform), the score represents each feature based on how important it is to the client (see Fig. 1). The rating for a feature is calculated from review sets in one of two ways:

- (1) for reviews of a certain class, in which reviewers give stars or otherwise numerically (or ordinally) rank products according to certain classes, the ratings serve as a set of utility weights for these data points;
- (2) for reviews that do not include numerical or ordinal values, the documents are processed with a heuristic that attempts to assign such ratings based on keywords or anti-keywords that appear in the text.

For instance, if "fast learning curve" is used to describe the product, it is a positive indicator for the *ease of use* feature of the product, while "buggy product" would be a negative indicator for the *stability* feature. The formula used to calculate the overall score of a product is as follows:

$$\text{overall\_score} = \text{price\_score} * \text{price\_weight} + \text{quality\_score} * \text{quality\_weight} \\ + \text{hardware\_score} * \text{hardware\_weight}$$

Since the information comes from different sources, there may be inconsistencies, and different sources may have different relative quality or confidence measures. The value of information in our system is determined by the value of the source; information from a high-quality source is considered to be closer to the truth. To combine inconsistent information from different sites, we classify information sources as one of three categories: high, medium or low quality. The classification of a known information source is based on human knowledge and prior experience about this source. Our rating system for unknown sources currently employs a URL reference search to rank sites, similar to the Usenet-based approach outlined in [58]. Several Web search engines offer a service which allows the user to search for Web sites which link to a certain page. This essentially allows us to quantify how often a Web site is referenced by others. Our heuristic ranks sites based on the assumption that sites which are more useful and credible will be referenced more often than less credible ones. This trait offers two important qualities: it is independent,

since the rating is not dictated by any one person or company, and it is also quite generic. Although we do not do so, it would also be possible to augment this rating with either user feedback (e.g., “I typically don’t agree with this reviewer’s point of view”) or data from a centralized Web rating service. This rating is also used in the initial selection of which documents to process. For each kind of information source, there is a quality measure distribution table that describes the relationship between the information from this source and the possible truth values. These quality measure distribution tables were constructed in an ad hoc fashion, based on hand reviewing of documents from different categories and experience running the system. These tables help provide more accurate interpretations of data from those information sources, by taking the observed rating and transposing it into a distribution of possible ratings, weighted by the source’s quality. The net effect of this mapping is to add some measure of skepticism to the system, based on the quality of the site. If, for instance, a highly rated site gives a certain product a rating of 5, BIG is more apt to believe that the review is accurate, as compared to a lower rated site which offers the same product rating.

For example, the product “Corel WordPerfect”, based on a review from site A, is highly rated (it is given a product quality of 4). The review from site B gives it a slightly lower rating of 3. Site A itself is known to be a medium-quality site with a source quality of 2, while B has a higher quality rating of 3. The quality measure tables for sites A and B are shown in Figs. 13, and 14, respectively.

| Observed<br>Review Quality | Interpreted Quality Distribution |            |            |            |            |
|----------------------------|----------------------------------|------------|------------|------------|------------|
|                            | 5                                | 4          | 3          | 2          | 1          |
| 5                          | 0.1                              | 0.2        | 0.5        | 0.1        | 0.1        |
| <b>4</b>                   | <b>0.0</b>                       | <b>0.3</b> | <b>0.2</b> | <b>0.3</b> | <b>0.2</b> |
| 3                          | 0.0                              | 0.1        | 0.3        | 0.3        | 0.3        |
| 2                          | 0.0                              | 0.0        | 0.5        | 0.5        | 0.3        |
| 1                          | 0.0                              | 0.0        | 0.3        | 0.6        | 0.6        |

Fig. 13. Review quality interpretation table for site A (source quality 2).

| Observed<br>Review Quality | Interpreted Quality Distribution |            |            |            |            |
|----------------------------|----------------------------------|------------|------------|------------|------------|
|                            | 5                                | 4          | 3          | 2          | 1          |
| 5                          | 0.7                              | 0.2        | 0.1        | 0.0        | 0.0        |
| 4                          | 0.2                              | 0.6        | 0.2        | 0.0        | 0.0        |
| <b>3</b>                   | <b>0.0</b>                       | <b>0.2</b> | <b>0.7</b> | <b>0.1</b> | <b>0.0</b> |
| 2                          | 0.0                              | 0.0        | 0.2        | 0.7        | 0.1        |
| 1                          | 0.0                              | 0.0        | 0.0        | 0.3        | 0.7        |

Fig. 14. Review quality interpretation table for site B (source quality 3).

Based on the review quality from site A and site B and their quality measures, the decision maker gets *quality\_score* distribution as: [(4, 0.25) (3, 0.45) (2, 0.2) (1, 0.1)]. This means there is 25% probability that quality of “Corel WordPerfect” is 4, 45% probability it is 3, 20% probability it is 2, and 10% probability it is 1. The expected *quality\_score* of “Corel WordPerfect” is therefore 2.85. Thus, for each product, the decision maker has a *quality\_score* distribution and an expected score. The product with the highest expected score is recommended to the client and the score distributions are used to calculate the confidence of this decision.

In addition to the decision and product information, the agent also gives the evaluation of this decision to the client. Since there are many factors contributing to the evaluation of the decision, it is difficult to represent the decision evaluation as a single number. We chose decision coverage and precision as two main characteristics of the decision.

Decision coverage is a 3-dimensional vector:

- (1) *Total product number*. Indicates how many products the agent has found; the more products the agent finds, the higher the quality of the decision.
- (2) *Candidate product number*. Describes the number of competing products used as candidates in final decision; the more products that are considered for the decision, the higher the quality of the decision.
- (3) *Information coverage*. Reflective of the number of documents the agent has processed.

Decision precision is a 4-dimensional vector:

- (1) *Average coverage*. Indicates the average number of documents supporting each candidate product.
- (2) *Information quality*. Describes the distribution of high-quality sources, medium-quality sources, and low-quality sources respectively.
- (3) *Process accuracy*. Measures how accurately the agent processes documents. Since the information extraction process is not perfect for any document, the extraction tool provides the degree of belief for every item it returns. For example, texttext-ks may find the operating system for “Corel WordPerfect” is “mac”, with a degree of belief of 0.8. The process accuracy is the average of the degree of belief of all items.
- (4) *Decision confidence*. Measures how confident the agent feels that the product it recommended to the client is the best product it found. This is computed from the quality distributions of the discovered products. For example, if product A has a distribution of [(5, 0.3) (4, 0.6) (3, 0.1)], product B has score distribution [(5, 0.1) (4, 0.3) (3, 0.3) (2, 0.2)], product A is recommended because it has a higher expected score. The possibility B is better than A is:  $0.1 * (0.6 + 0.1) + 0.3 * (0.1) = 0.1$ , so the confidence of this decision is  $1 - 0.1 = 0.9$ ;

Using this decision evaluation data allows the client to analyze the final decision with a more critical eye. An additional tool that we have not yet implemented is an appropriate interface for a client to access the raw data that was used by the system in making its decision.

### 5.6. Performance under varying time constraints

Table 3 illustrates how the system operates under different time constraints. The experiments cover searches looking for word processing products. The search and product criteria is the same for all runs, only the time allotted for the search varies. The intent of this section is to show that the system can effectively exploit the time allocated to it by the user to complete its search, and that in most cases its intended schedule closely approximates the actual execution time.

The first four columns of data provide information about the duration of each search. *User Time* denotes the user's target search time; the value in parentheses represents the upper bound on how far over the target search time the scheduler was permitted to go in order to achieve a good quality/cost/duration trade-off. (Utility in these cases is linearly decreasing between the expressed deadline and 10% above the expressed deadline.<sup>17</sup>) *Scheduled* denotes the expected total duration of the schedule produced by the Design-to-Criteria scheduler and *Execution* denotes the actual duration of the discovery and decision process. The difference in these values stems from the high variance of Web-related activities and reflects issues like changes in network bandwidth during the search, slowdowns at remote sites, and so forth. The statistical characterizations of these activities are also often imperfect, though they are improved over time. Given the variances involved, we are satisfied with the relationship between expectations and reality.

The next four columns denote number of considered products (#C.P.), total number of products found (T.P.), aggregate information coverage (I.C.), and average information coverage per product object (A.C.). These values reflect the number and qualities of the information sources used to generate the final decision. Given additional time, 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 results of this behavior can be seen in the correlation between longer running time and larger information coverage values; these values represent the total number of documents found and the average number of supporting documents a product has, respectively. As one would expect, the larger number of information sources also serves to increase both the number of known products and the size of the subset selected for consideration, which in turn affects the confidence BIG has in its final decision.

The last two columns describe how confident the system is in the information extraction and decision-making processes. Process accuracy (P.A.), supplied in part by the information processing tools, is the degree of belief that the actual extracted information is correctly categorized and placed in the information objects. Decision confidence (D.C.), 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 decision confidence therefore is not dependent on execution time or process accuracy.

---

<sup>17</sup> This approach to deadlines was taken to address client preferences. Despite requests to use a hard deadline model, clients were often dissatisfied if much better results were possible for slightly more time, and the scheduler selected an option that stayed within the expressed deadline.

Table 3  
Different time allotments produce different results

| User Time | #  | Scheduled | Execution | #C.P. | #T.P. | I.C. | A.C. | P.A. | D.C. |
|-----------|----|-----------|-----------|-------|-------|------|------|------|------|
| 300(330)  | 1  | 311       | 367       | 4     | 10    | 12   | 1.5  | 1.6  | 1    |
|           | 2  | 308       | 359       | 3     | 10    | 16   | 1.3  | 1.4  | 1    |
|           | 3  | 305       | 279       | 3     | 10    | 11   | 1.3  | 1.5  | 1    |
|           | 4  | 311       | 275       | 3     | 11    | 13   | 1.67 | 1.5  | 1    |
|           | 5  | 321       | 286       | 4     | 10    | 12   | 1.5  | 1.6  | 1    |
|           | 6  | 321       | 272       | 3     | 10    | 12   | 1.3  | 1.6  | 0.84 |
|           | 7  | 262       | 327       | 3     | 11    | 12   | 1.67 | 1.5  | 1    |
|           | 8  | 262       | 337       | 3     | 10    | 11   | 1.3  | 1.5  | 1    |
|           | 9  | 262       | 301       | 2     | 11    | 10   | 1.0  | 1.4  | 1    |
|           | 10 | 259       | 292       | 2     | 11    | 11   | 1.5  | 1.5  | 1    |
| average   |    | 302       | 310       | 3     | 10.4  | 12   | 1.4  | 1.5  | 0.98 |
| s.d.      |    | 33        | 35        | 0.67  | 0.5   | 1.6  | 0.2  | 0.07 | 0.05 |
| 600(660)  | 1  | 658       | 760       | 6     | 17    | 45   | 4.0  | 1.7  | 0.99 |
|           | 2  | 658       | 608       | 4     | 17    | 44   | 6.75 | 1.8  | 1.0  |
|           | 3  | 645       | 732       | 5     | 20    | 46   | 5.4  | 2    | 1.0  |
|           | 4  | 649       | 809       | 10    | 28    | 49   | 3.1  | 1.8  | 0.96 |
|           | 5  | 649       | 730       | 7     | 17    | 42   | 4.3  | 1.8  | 0.84 |
|           | 6  | 653       | 774       | 4     | 23    | 55   | 6.5  | 2.3  | 0.99 |
|           | 7  | 653       | 671       | 4     | 18    | 35   | 5.3  | 2.1  | 0.99 |
|           | 8  | 653       | 759       | 6     | 18    | 41   | 4.8  | 2.2  | 0.84 |
|           | 9  | 653       | 760       | 5     | 28    | 50   | 5.4  | 2.2  | 0.94 |
|           | 10 | 653       | 852       | 5     | 18    | 42   | 4.6  | 2.0  | 0.85 |
| average   |    | 652       | 746       | 5.6   | 20    | 45   | 5.0  | 2.0  | 0.95 |
| s.d.      |    | 4         | 68        | 1.8   | 4.4   | 5.6  | 1.1  | 0.2  | 0.06 |
| 900(990)  | 1  | 951       | 975       | 5     | 37    | 61   | 5.8  | 2.2  | 0.99 |
|           | 2  | 968       | 956       | 8     | 30    | 55   | 4.1  | 2.1  | 1    |
|           | 3  | 914       | 919       | 8     | 23    | 64   | 4.0  | 1.9  | 1    |
|           | 4  | 960       | 796       | 6     | 34    | 64   | 5.3  | 1.9  | 0.96 |
|           | 5  | 960       | 1026      | 9     | 24    | 32   | 4.1  | 1.9  | 0.99 |
|           | 6  | 987       | 968       | 8     | 27    | 60   | 4.4  | 2.1  | 0.94 |
|           | 7  | 987       | 1102      | 8     | 27    | 63   | 5.5  | 1.7  | 0.94 |
|           | 8  | 987       | 896       | 5     | 32    | 69   | 5.4  | 2.1  | 0.84 |
|           | 9  | 987       | 918       | 7     | 32    | 66   | 5.1  | 2.0  | 0.84 |
|           | 10 | 978       | 1289      | 14    | 39    | 79   | 3.9  | 2.0  | 1    |
| average   |    | 968       | 985       | 7.8   | 31    | 61   | 4.8  | 2.0  | 0.95 |
| s.d.      |    | 23        | 134       | 2.6   | 5.3   | 12   | 0.7  | 0.14 | 0.06 |

Key: User Time = user’s preferred search time (linearly decreasing utility post- deadline in this case), Scheduled = total execution time as predicted by model and anticipated by scheduler, Execution = actual execution time, I.C. = information coverage, T.P. = total product objects constructed, #C.P. = total products passed to decision process, A.C. = average coverage per object, P.A. = extraction processing accuracy per object, D.C. = overall decision process confidence, s.d. = standard deviation.

Our query for the test runs is that of a client looking for a word processing package for the Macintosh costing no more than \$200, and would like the search process to take 300/600/900 seconds and the search cost to be less than five dollars. The client specifies the relative importance of price to quality to be 60/40 and the relative importance of coverage to confidence to be 50/50.

Looking at the results, one can see that the process accuracies for the 300-second run are consistently lower than those for the 600- and 900-second runs, which are roughly the same. Process accuracy is affected by the amount of available evidence, in that matching information from different sources increases the perceived accuracy of the data. Since the latter two runs have similar average coverage values, one would expect similar levels of information matching, and thus similar levels of process accuracy. Using the same logic, one can see why the process accuracy for the 300-second runs would be consistently lower, resulting from its lower levels of average coverage.

The decision confidence value is affected by both the number of products considered and their respective attributes and qualities. BIG first selects a product, based on its attributes and the user's preferences. It then calculates the decision confidence by determining the probability that the selected product is the optimal choice, given the available subset of products. In the 300-second runs, the total number of considered products is fairly low, which increases the chance that the pool of products is heterogeneous. In such a population, it is more likely that a single candidate will stand out from the others, which goes to explain the large percentage of perfect scores in the shortest run. When BIG is given more time to find more products, the chance that individual candidates will sharply contrast is reduced. Greater average coverage affects this contrast by increasing the likelihood that product candidates will be fully specified. This will typically make the candidate set have a higher quality rating which makes the population more homogeneous. It is this blurring across attribute dimensions which reduces BIG's confidence in the final decision.

Two interesting cases in this last column are worth explaining in more detail. In the sixth 300-second run, one can see that the decision quality was calculated to be 0.84, much lower than other runs in the same set. This was due to the fact that two of the three products considered were actually the same product, but one was an academic version. These two products had relatively similar quality ratings, which were significantly higher than the remaining product, which caused BIG to have a lower confidence in its decision. The second anomaly occurs in the tenth run in the 900-second scenario. In this case, 14 products were considered for selection. Of the group, 11 had a price higher than \$400, two were above \$200 and the remaining product was roughly \$70 with good coverage of the user's desired characteristics. This large price discrepancy led the selected product to have a much higher quality rating than the competition, which led to the high decision confidence.

### 5.7. Experiments in different domains

Besides the word processing domain, we have also experimented with BIG in three other domains: image editors, html editors, and database systems. For each domain, we spent approximately one hour collecting and classifying documents to form a corpus for



| domain       |         | Scheduled | Execution | #C.P. | #T.P. | I.C. | A.C. | P.A. | D.C. | final decision  |
|--------------|---------|-----------|-----------|-------|-------|------|------|------|------|---|
| image editor | average | 541       | 520       | 4     | 7     | 28.2 | 5.3  | 0.8  | 1    | product name: Adobe Image Ready                                     |
|              | s.d.    | 0         | 8         | 0     | 0     | 1.9  | 0.5  | 0    | 0    | platform: windows_nt, mac, win95<br>price: 188.98 occurrence: 10/10 |
| html editor  | average | 541       | 595       | 8     | 12    | 49.1 | 5.1  | 0.9  | 0.84 | product name: Adobe Pagemill 3.0                                    |
|              | s.d.    | 0         | 59        | 0.6   | 0     | 3.8  | 0.7  | 0.05 | 0.04 | platform: windows_nt, mac, win95<br>price: 79.98 occurrence: 8/10   |
| database     | average | 541       | 548       | 4.6   | 10.2  | 43.6 | 8    | 1    | 0.97 | product name: FileMaker Pro 4.0                                     |
|              | s.d.    | 0         | 19        | 0.8   | 1.17  | 5.2  | 1.8  | 0.06 | 0.07 | platform: win95, mac, windows_3.1<br>price: 159.98 occurrence: 7/10 |

Key: Scheduled = total execution time as predicted by the model and anticipated by the scheduler, Execution = actual execution time, #C.P. = total candidate products passed to the decision process, # T.P. = total product objects constructed, I.C. = information coverage, A.C. = average coverage per object, P.A. = extraction processing accuracy per object, D.C. = overall decision process confidence; final decision = the product most frequently recommended by the system in 10 runs; occurrence = the frequency the product recommended by the system; s.d. = standard deviation.

Fig. 15. Experiments in three different domains.

the Bayes classifier. As we discussed in Section 5.1, this process can be automated by equipping BIG with the ability to learn from user feedback.

Fig. 15 shows the results of experiments in each of these domains. The query criteria for each domain specifies that a relevant software package for the Windows platform is needed, with a search deadline of 10 minutes (600 seconds); the query was repeated 10 times for each domain. All items in Fig. 15 also appear in Table 3 and have been explained in Section 5.6. The DTC scheduler generates the same schedule for all queries because the search and product criteria are the same and the object database is cleared after each run (BIG does not have knowledge about these products *a priori* and knowledge that is learned during one trial is removed before the next trial).

The search for html editor products takes longer to execute than the other two queries because there are more available products (#C.P. and #T.P.) in this domain. Thus some methods take longer time than expected, because their execution times are related to the number of product descriptions that are either retrieved or actively being considered as candidates (such as the “Search\_For\_Reviews” method).

The final decision presents the product that is most frequently recommended by the system in the 10 runs. The occurrence indicates how many times it is recommended in those 10 runs. Different products are recommended in different runs because the system does not have sufficient time to exhaustively process all available documents, so it randomly selects equally rated documents for processing. The selected document set therefore varies from one run to the next, so the system may have different information, which in turn may support an alternate final decision.

For the image editor domain, the Adobe product Image Ready was recommended each time because there are fewer candidate products in this domain and because Image Ready is the best among the candidate products; other products’ prices are about \$300, which is much higher than the price of Image Ready. For the html editor domain, out of 10 runs, the system recommended Adobe Pagemill 3.0 eight times, and in the other two runs it recommended Symantec’s Visual Page 2.0 and WebSpice respectively. All three products have similar functionality and price. In the database domain, the system recommended FileMaker Pro seven times, MS Access once and MSFT Access for Win 95 Step by

Step twice. The first two products are reasonable selections, but the last one is actually a book. Our classifier fails to reject this alternative since there were many database-related keywords in the description and it had a very low price compared with other real software products.<sup>18</sup>

The techniques and technologies employed by BIG are also applicable to decision-making tasks in other domains as well, such as the car purchasing domain mentioned earlier. The core components of BIG, including the RESUN planner, design-to-criteria scheduler, task assessor, and databases are entirely domain independent, and may be used without modification to address different questions. The blackboard architecture is also domain independent, but a designer would need to enumerate different characteristics for objects placed on the blackboard. In the car domain, for instance, the “hard drive” and “platform” traits are not pertinent; elements such as “engine” or “model” would be more appropriate. The most demanding aspects requiring changes in a new domain relate to text processing. Both the NLP-style text extraction (texttext-ks) and the document classifier rely on a one-time training session on a domain corpus to achieve their results. The wrapper text extraction utility (quickext-ks) is also domain dependent, and would require new rules to correctly process the different Web sites used in the domain. The other text extractors (grep-ks, cgrep-ks, tablext-ks) are domain independent.

## 6. Strengths, limitations, and future directions

The combination of the different AI components in BIG and the view of information gathering as an interpretation task has given BIG some very strong abilities. In contrast to most other work done in this area, BIG performs *information fusion*—not just document retrieval. That is, BIG retrieves documents, extracts attributes from the documents, converting unstructured text to structured data, and integrates the extracted information from different sources to build a more complete model of the product in question. The use of the RESUN interpretation-style planner enables BIG to reason about the sources-of-uncertainty associated with particular aspects of product objects and to plan to resolve these uncertainties by gathering and extracting more information that serves as either corroborating or negating evidence. Though this feature was not brought out in our simple trace, it is a definite strength when operating in a realm of uncertain information combined with uncertain extraction techniques.

BIG is also quite scalable, because of the way it can filter and focus a large amount of information into a concise, useful representation. BIG can obtain and process information from a large variety of sources on the Web, and given sufficient time, can process as much information as it is able to find during its search. The hierarchical structure of the blackboard allows BIG to effectively use the data because it is condensed and abstracted as it rises through the levels. The key phrase here, however, is *given sufficient time*.

---

<sup>18</sup> There is an interesting question whether more extensive training of the classifier would have solved this problem. We could have also added special heuristics to reflect the fact that products priced so low for the genre were likely either to not be complete software packages or not software. Shareware, freeware and other low-cost, but potentially viable, solutions further complicate issues.

BIG is still a single entity system, so while the higher levels of the blackboard are manageable, the lower levels can accumulate a large number of objects on them, which can be slow to process. A proposed solution to this problem is to make BIG a multi-agent system. In such a system, multiple BIG agents could independently search for information, conceptually branching the lower levels of the blackboard. Not only will this solution increase information throughput through parallelism, but communication between the agents can also help focus the system as a whole.

Another feature of BIG not fully detailed in this paper is the use of the Design-to-Criteria scheduler to reason about the quality, cost, time, and certainty trade-offs of different candidate actions. The use of the scheduler enables BIG to address deadlines and search resource constraints, a feature that is particularly important given the scope of the search space, the uncertainty involved, and the very real requirement for information systems to address time and resource constraints. Relatedly, while the issue of planning for information cost constraints is not stressed in this paper, we feel that in the future the cost of accessing particular information sources will need to be taken into account by information gathering agents. Examples of the use of cost in BIG's IG process are presented in [45].

Also not a focus of this paper is BIG's ability to learn from prior problem-solving instances [43]. Information objects (along with their associated sources-of-uncertainty) can be stored and used to supplement subsequent search activities. In this fashion, BIG gains from prior problem-solving instances, but, it also refines and modifies the product models over time by resolving previously unresolved SOUs and gathering new information about the products.

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; 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, as discussed in Section 5.7, information extractors like the information extraction system, *cgrepxt-ks*, and *quickext-ks*, require supervised training to learn extraction rules and make use of semantic dictionaries to guarantee a certain level of performance. 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.

Another possible limitation with the current incarnation of BIG is the use of text extraction technology to convert unstructured text to structured data. The text extraction techniques are sometimes fragile, particularly when asked to extract data from a document not belonging to the class of document on which the system was trained. The use of a document classifier greatly improves the situation, but, information extraction remains a nontrivial issue. The use of XML and other data structuring mechanisms on the Web will help alleviate this issue. Interestingly, because RESUN represents and works to resolve sources-of-uncertainty, the limitations and sometimes erroneous output of the text extraction tools is not nearly as problematic as it might seem at first glance. Given sufficient time for search, the planner will usually recover from misdirections stemming from poor information extraction.

Our future interests lie in improving the integration of the top-down view of the Design-to-Criteria Scheduler and the opportunistic bottom-up view of the RESUN planner. Currently, the scheduler's primary role in the system is to produce the initial schedule. However, as the control structure evolves, we foresee a feedback loop in which the RESUN planner and the task assessor pose what-if type questions to the scheduler to support high-level decisions about which actions to perform next. A stronger two-way interface will also support more opportunistic problem-solving strategies by enabling the problem solver to respond to changes and evaluate the value of changing its planned course of action. We see this as particularly valuable in light of the uncertainty in the information gathering domain and the high-order combinatorics of the trade-off decision process. In this secondary role, the scheduler becomes the trade-off expert employed by the task assessor/problem solver to guide agent activities during execution. Another important direction is to exploit user feedback about the appropriateness of the system's decision and the documents/sites that support that decision. We feel this will allow us to both strengthen the processing of existing software genres and also allow us to more seamlessly integrate other software genres that have not been trained for. Finally, we would like to move BIG into a multi-agent system involving mobile agents. Our group [60] has a long history of developing distributed problem-solving and multi-agent systems [9,17,18,37,40,54] and we are interested in exploring multi-agent coordination via a group of agents descended from the current BIG agent.

### **Acknowledgements**

We would like to thank the reviewers for their thoughtful comments and editing suggestions that served to improve the overall quality of this paper.

Because of the large number of people involved with this effort we would like to acknowledge and identify the contributions of the authors and other individuals. Authors are listed alphabetically, except for the PI.

Victor Lesser had overall responsibility for the architecture definition and provided project management throughout the effort. Bryan Horling developed the Web interface for simple document retrieval and search engine usage, implemented the server information and object databases, implemented quickext-ks (the wrapper-style information extraction utility), created the Web spider that supplements BIG's other knowledge sources, and constructed the GUI for BIG. Frank Klassner contributed by facilitating the integration of the RESUN planner, implementing the blackboard data structures, and by providing project guidance in the early stages of the work. Anita Raja focused primarily on the extraction issues and implemented the grep-ks, cgrep-ks, and table-ks knowledge sources. Raja also modified the BADGER extraction system to be compatible with BIG's domain, enhanced BADGER's back-end processing, and integrated the document classifier into the system. Shelley XQ. Zhang designed and implemented the task assessor and the decision maker, constructed RESUN's control plans, constructed the interface between RESUN and the Design-to-Criteria Scheduler, and contributed to the rest of the system integration. Zhang also handled all of the experimental work and was primarily responsible for the project and modifications during its later stages. Tom Wagner contributed to the project management,

to the overall development of the architecture, to the selection and preliminary testing of the individual components of BIG, and provided TÆMS and scheduling support.

We would also like to thank Professor Norman Carver for his original development of the RESUN planner used in BIG and his contributions to the interface between the planner and the scheduler. The ideas behind the task assessor are partly attributable to Carver as well.

Jonathan Aseltine, formerly of the UMASS NLP group, also deserves credit for his assistance in the modification of the BADGER information extraction system. Thanks also to David Fisher, also of the NLP group, for his time in making the BADGER system available to us.

In terms of the intellectual underpinnings of this project, thanks are also in order to Mike Chia for his thoughts on the application and hand generated proof-of-concepts, as well as to Tim Oates and M.V. Nagendra Prasad who, along with Victor Lesser, developed the foundational multi-agent scenario [50] of this application in 1994. Appreciation also goes to Keith Decker and Tom Wagner for their efforts on the pre-BIG prototypes and hand-run scenarios and to Mia Stern, who, along with Tom Wagner, studied the state of the Internet and agent information gathering and provided intellectual support for the BIG objectives in 1994.

Thanks also to Peter Amstutz and Kyle Rawlins, undergraduates, for their work that was not used in this article and to Manik Ahuja, also an undergraduate, who did a portion of the preliminary text mark-up work.

## References

- [1] S. Adali, K.S. Candan, Y. Papakonstantinou, V.S. Subrahmanian, Query processing in distributed mediated systems, in: Proc. 1996 ACM SIGMOD Conference on Management of Data, 1996.
- [2] ANSI/NISO z39.50-1992, American National Standard Information retrieval application service definition and protocol specification for open systems interconnection, 1992.
- [3] Y. Arens, C. Knoblock, W. Shen, Query reformulation for dynamic information integration, *J. Intelligent Information Systems (Special Issue on Intelligent Information Integration)* 6 (2–3) (1996) 99–130.
- [4] Autonomy agentware technology white paper. <http://www.agentware.com/main/tech/whitepaper.htm>.
- [5] C.M. Bowman, P.B. Danzig, U. Manber, M.F. Schwartz, Scalable internet resource discovery: Research problems and approaches, *Comm. ACM* 37 (8) (1994) 98–114.
- [6] J.P. Callan, W. Bruce Croft, S.M. Harding, The INQUERY retrieval system, in: Proc. 3rd International Conference on Database and Expert Systems Applications, 1992, pp. 78–83.
- [7] N. Carver, V. Lesser, The evolution of blackboard control architectures, *Expert Systems with Applications (Special Issue on The Blackboard Paradigm and Its Applications)* 7 (1) (1994) 1–30.
- [8] N. Carver, V. Lesser, A planner for the control of problem solving systems, *IEEE Trans. Systems Man Cybernet (Special Issue on Planning, Scheduling and Control)* 6 (1993) 1519–1536.
- [9] N. Carver, V. Lesser, The DRESUN testbed for research in FA/C distributed situation assessment: Extensions to the model of external evidence, in: Proc. First International Conference on Multiagent Systems, 1995.
- [10] W. Clancey, From GUIDON to NEOMYCIN and HERACLES in twenty short lessons, *AI Magazine* 7 (3) (1986) 40–60.
- [11] W. Clancey, C. Bock, Representing control knowledge as abstract tasks and metarules, Technical Report KSL 85-16, Knowledge Systems Laboratory, Computer Science Department, Stanford University, 1986.
- [12] J. Cowie, W.G. Lehnert, Information extraction, *Comm. ACM* 39 (1) (1996) 80–91.
- [13] T. Dean, M. Boddy, An analysis of time-dependent planning, in: Proc. AAAI-88, St. Paul, MN, 1988, pp. 49–54.

- [14] K.S. Decker, Environment centered analysis and design of coordination mechanisms, Ph.D. Thesis, University of Massachusetts, 1995.
- [15] K.S. Decker, Task environment centered simulation, in: M. Prietula, K. Carley, L. Gasser (Eds.), *Simulating Organizations: Computational Models of Institutions and Groups*, AAAI Press/MIT Press, Cambridge, MA, 1996.
- [16] K.S. Decker, V.R. Lesser, Quantitative modeling of complex environments, *Internat. J. Intelligent Systems in Accounting, Finance, and Management (Special Issue on Mathematical and Computational Models of Organizations: Models and Characteristics of Agent Behavior)* 2 (4) (1993) 215–234.
- [17] K.S. Decker, V.R. Lesser, Designing a family of coordination algorithms, in: *Proc. First International Conference on Multi-Agent Systems*, San Francisco, CA, AAAI Press, Menlo Park, CA, 1995, pp. 73–80. Longer version available as UMass CS-TR 94–14.
- [18] K.S. Decker, M.V. Nagendra Prasad, T. Wagner, MACRON: An architecture for multi-agent cooperative information gathering, in: *Proc. CIKM-95 Workshop on Intelligent Information Agents*, Baltimore, MD, 1995.
- [19] K. Decker, A. Pannu, K. Sycara, M. Williamson, Designing behaviors for information agents, in: *Proc. 1st Internat. Conference on Autonomous Agents*, Marina del Rey, CA, 1997, pp. 404–413.
- [20] R. Doorenbos, O. Etzioni, D. Weld, A scalable comparison-shopping agent for the World-Wide-Web, in: *Proc. First International Conference on Autonomous Agents*, Marina del Rey, CA, 1997, pp. 39–48.
- [21] O. Etzioni, Moving up the information food chain: Employing softbots on the World Wide Web, in: *Proc. AAAI-96*, Portland, OR, 1996, pp. 1322–1326.
- [22] O. Etzioni, S. Hanks, T. Jiang, R. Karp, O. Madani, O. Waarts, Optimal information gathering on the internet with time and cost constraints, in: *Proc. 37th IEEE Symposium on Foundations of Computer Science (FOCS)*, Burlington, VT, 1996.
- [23] T. Finin, R. Fritzson, D. McKay, R. McEntire, KQML as an agent communication language, in: *Proc. 3rd International Conference on Information and Knowledge Management, CIKM'94*, ACM Press, New York, 1994.
- [24] D. Fisher, S. Soderland, J. McCarthy, F. Feng, W. Lehnert, Description of the UMass systems as used for MUC-6, in: *Proc. 6th Message Understanding Conference*, Columbia, MD, 1996.
- [25] J. Grass, S. Zilberstein, Value-driven information gathering, in: *Proc. AAAI Workshop on Building Resource-Bounded Reasoning Systems*, Providence, RI, 1997.
- [26] L. Haas, D. Kossmann, E. Wimmers, J. Yang, Optimizing queries across diverse data sources, *VLDB*, 1997.
- [27] J. Hammer, H. Garcia-Molina, K. Ireland, Y. Papakonstantiou, J. Ullman, J. Widom, Information translation, mediation, and Mosaic-based browsing in the TSIMMIS system, in: *Proc. ACM SIGMOD International Conference on Management of Data*, 1995.
- [28] W. Hasan, D. Florescu, P. Valduriez, Open issues in parallel query optimization, *SIGMOD Record* 3 (1996) 28–33.
- [29] E. Horvitz, G. Cooper, D. Heckerman, Reflection and action under scarce resources: Theoretical principles and empirical study, in: *Proc. IJCAI-89*, Detroit, MI, 1989.
- [30] E. Horvitz, J. Lengyel, Flexible rendering of 3D graphics under varying resources: Issues and directions, in: *Proc. AAAI Symposium on Flexible Computation in Intelligent Systems*, Cambridge, MA, 1996.
- [31] A. Howe, D. Dreilinger, A meta-search engine that learns which engines to query, *AI Magazine* 18 (2) (1997).
- [32] <http://macworld.zdnet.com/pages/april.97/reviews.3334.html>.
- [33] Inforia quest, <http://www.inforia.com/quest/iq.htm>.
- [34] Jango, <http://www.jango.com/>.
- [35] P.D. Karp, A strategy for database interoperation, *J. Comput. Biology* 4 (1995) 573–583.
- [36] B. Krulwich, The BargainFinder agent: Comparison price shopping on the Internet, in: J. Williams (Ed.), *Bots and Other Internet Beasties*, SAMS.NET, 1996. <http://bf.cstar.ac.com/bf/>.
- [37] S. Lander, V.R. Lesser, Negotiated search: Organizing cooperative search among heterogeneous expert agents, in: *Proc. 5th International Symposium on Artificial Intelligence Applications in Manufacturing and Robotics*, 1992.
- [38] L. Larkey, W. Bruce Croft, Combining classifiers in text categorization, in: *Proc. 19th International Conference on Research and Development in Information Retrieval (SIGIR '96)*, Zurich, Switzerland, 1996, pp. 289–297.

- [39] W.G. Lehnert, B. Sundheim, A performance evaluation of text analysis technologies, *AI Magazine* 12 (3) (1991) 81–94.
- [40] V.R. Lesser, Reflections on the nature of multi-agent coordination and its implications for an agent architecture, *Autonomous Agents and Multi-Agent Systems* 1 (1) (1998) 89–111.
- [41] V. Lesser, S. Zilberstein, Intelligent information gathering for decision models, Computer Science Technical Report TR-96-35, University of Massachusetts at Amherst, 1996.
- [42] V. Lesser, M. Atighetchi, B. Horling, B. Benyo, A. Raja, R. Vincent, T. Wagner, P. Xuan, S.XQ. Zhang, A multi-agent system for intelligent environment control, in: Proc. 3rd International Conference on Autonomous Agents (Agents99), Seattle, WA, 1999.
- [43] V. Lesser, B. Horling, F. Klassner, A. Raja, T. Wagner, S.XQ. Zhang, BIG: A resource-bounded information gathering agent, in: Proc. AAAI-98, Madison, WI, 1998. See also UMass CS Technical Reports 98-03 and 97-34.
- [44] V. Lesser, B. Horling, F. Klassner, A. Raja, T. Wagner, S.XQ. Zhang, A next generation information gathering agent, in: Proc. 4th International Conference on Information Systems, Analysis, and Synthesis, 1998, pp. 41–50. In conjunction with the World Multiconference on Systemics, Cybernetics, and Informatics (SCI'98), Vol. 2. Also available as UMass CS TR 1998-72.
- [45] V. Lesser, B. Horling, A. Raja, T. Wagner, S.XQ. Zhang, Sophisticated information gathering in a marketplace of information providers, Computer Science Technical Report TR-99-57, University of Massachusetts at Amherst, 1999. Under review.
- [46] A. Levy, A. Rajaraman, J.J. Ordille, Query-answering algorithms for information agents, in: Proc. AAAI-96, Portland, OR, 1996, pp. 40–47.
- [47] I. Musela, S. Minton, C. Knoblock, A hierarchical approach to wrapper induction, in: Proc. 3rd Annual Conference on Autonomous Agents (Agents99), Seattle, WA, ACM Press, New York, 1999, pp. 190–197.
- [48] I. Muslea, S. Minton, C. Knoblock, STALKER: Learning extraction rules for semistructured, Web-based information sources, in: Proc. AAAI Workshop on AI and Information Integration, 1998.
- [49] Naive Bayes document classifier, Supplement to Machine Learning by T. Mitchell, McGraw Hill, New York, 1997. <http://www.cs.cmu.edu/afs/cs/project/theo-11/www/naive-bayes.html>.
- [50] T. Oates, M.V. Nagendra Prasad, V.R. Lesser, Cooperative information gathering: A distributed problem solving approach, Computer Science Technical Report 94-66, University of Massachusetts, 1994; also: *Journal of Software Engineering (Special Issue on Developing Agent Based Systems)* (1997).
- [51] C. Rich, C.L. Sidner, Collagen: When agents collaborate with people, in: Proc. 1st International Conference on Autonomous Agents (Agents97), Marina del Rey, CA, 1997, pp. 284–291.
- [52] Robo surfer, <http://www.robosurfer.com/>.
- [53] S.J. Russell, S. Zilberstein, Composing real-time systems, in: Proc. IJCAI-91, Sydney, Australia, 1991, pp. 212–217.
- [54] T. Sandholm, An implementation of the contract net protocol based on marginal cost calculations, in: Proc. AAAI-93, Washington, DC, 1993, pp. 256–262.
- [55] S. Soderland, D. Fisher, J. Aseltine, W.G. Lehnert, Crystal: Inducing a conceptual dictionary, in: Proc. IJCAI-95, Montreal, Quebec, 1995, pp. 1314–1321.
- [56] R.R. Swick Ora Lassila, Resource description framework (RDF) model and syntax specification, Recommendation, World Wide Web Consortium, 1999. <http://www.w3.org/TR/PR-rdf-syntax/>.
- [57] TÆMS white paper, 1999. <http://mas.cs.umass.edu/research/taems/white/>.
- [58] L. Terveen, W. Hill, B. Amento, D. McDonald, J. Creter, A system for sharing recommendations, *Comm. ACM* 3 (1997) 59–62.
- [59] A. Tomasic, L. Raschid, P. Valduriez, A data model and query processing techniques for scaling access to distributed heterogeneous databases in disco, *IEEE Trans. Comput. (Special issue on Distributed Computing Systems)* (1997).
- [60] The Multi-Agent Systems Laboratory at the University of Massachusetts at Amherst, <http://mas.cs.umass.edu>.
- [61] T. Wagner, V. Lesser, Design-to-Criteria scheduling: Real-time agent control, Computer Science Technical Report TR-99-58, University of Massachusetts at Amherst, 1999. Under review.
- [62] T. Wagner, A. Garvey, V. Lesser, Complex goal criteria and its application in Design-to-Criteria scheduling, in: Proc. AAAI-97, Providence, RI, 1997, pp. 294–301. Also available as UMass CS TR-1997-10.

- [63] T. Wagner, A. Garvey, V. Lesser, Criteria-directed heuristic task scheduling, *Internat. J. Approx. Reason.* (Special Issue on Scheduling) 19 (1–2) (1998) 91–118. A version also available as UMASS CS TR-97-59.
- [64] M.P. Wellman, E.H. Durfee, W.P. Birmingham, The digital library as community of information agents, *IEEE Expert* (1996).
- [65] Zurfrider, <http://www.zurf.com/>.