BIG: A Resource-Bounded Information Gathering and Decision Support Agent *†

Victor Lesser   Bryan Horling   Frank Klassner   Anita Raja
Thomas Wagner   Shelley XQ. Zhang

Multi-Agent Systems Laboratory
Computer Science Department
University of Massachusetts
Amherst, MA 01003
Email: lesser@cs.umass.edu
http://mas.cs.umass.edu

UMass Computer Science Technical Report 1998-52

January 19, 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 coordination 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.

Keywords: Information Gathering, Information Agents, Information Systems, Planning, Scheduling, Text Processing.

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

* A version of this paper is also under review for a special issue of the AI Journal.

† 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.

information sources and their different levels of accessibility, reliability, completeness [3], 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 [4, 27, 7, 28] is no longer effective. The time/quality/cost tradeoffs offered by the collection of information sources and the dynamic nature of the environment lead us to conclude that the user cannot (and should not) serve as the detailed controller of the information gathering (IG) process.

Our solution to 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) [30, 31], 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. Implementationally, we have concentrated on the software domain and 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 constraints on the amount of money to pay for such a product and the amount of time and money to spend locating information about database products. The client may also specify a preference for information precision versus coverage, a coverage preference will result in more products being discovered, but with less information about each product. A preference for greater precision will result in BIG spending more resources to construct very accurate models of products. BIG will then plan, locate, and process 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 our information gathering agent's components. Indeed, several are complex problem solvers in their own right. A domain problem solver, a RESUN [5, 6] planner, translates a client's information need into a set of goals and generates plans to achieve those goals. To support reasoning about time/quality/cost trade-offs, and thus a range of different resource/solution paths, the planner enumerates multiple different ways to go about achieving the goals and describes them statistically in three dimensions, duration, quality, and cost, via discrete probability distributions. Another sophisticated problem solving component, a Design-to-Criteria [40] scheduler, examines the possible solutions paths and determines 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 schedule is a single-agent schedule that contains parallelism and overlapping executions 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 the performance of the parallel activities will not adversely affect the duration estimates associated with its activities.[1]

As BIG retrieves documents, yet another sophisticated problem solver, an IE system [17] in conjunction with a set of semantic, syntactic, and site specific tools, analyzes the unstructured text documents in order to construct information objects that can be used by the planner for decision making and refinement of other information gathering goals. It is important to note that the advent of widespread structure markup languages like XML, or tools to wrap web sites so that data can be retrieved in a structured form, will only improve BIG's ability to gather and process information. In essence, the IE system used in BIG fills the role

[1]The duration estimates associated with primitive actions are constructed assuming the dedicated efforts of the agent. In cases where multiple local activities are performed in parallel, the performance degradation will affect the duration estimates and result in schedules that miss deadlines and do not perform as expected.

of these other tools and approaches. In the event that structured information is available, the extraction step can be bypassed and the information incorporated directly into BIG's reasoning structures.

Other complex components in BIG include a framework for modeling domain tasks, a Web site information database, an idle-time Web site probe for refining the 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 in greater detail in Section 4.

The main 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.

Resource-boundedness BIG problem solves to meet real-time deadlines, cost constraints, and quality preferences. BIG 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.

Opportunistic and Top-down Control BIG blends opportunistic, reactive, problem solving behaviors with the end-to-end view required to meet real-time deadlines and other performance constraints. This enables BIG to respond dynamically to uncertainties in the product models as well as newly learned information.

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 in then incorporated and integrated into product models that are examined by BIG's decision process, resulting in a product recommendation.

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.

In Section 6.2 we present a detailed walk through of BIG in action, however, to preface the rest of the document and to illustrate some of BIG's capabilities, consider a high level example. A client is interested in finding a word processing program for a Macintosh. The client submits goal criteria that describes desired software characteristics and specifications for BIG's search-and-decide process. A snapshot of the systems's user specification form is given in Figure 1.

The search parameters are: *duration importance = 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 into emphasizing quality over duration and cost, a preference for a response in 10 minutes if possible, and a hard constraint that the search process cost no more than $5 (if information is purchased, as from the Consumers Reports website). 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, usefulness importance rating 50 units, future usefulness rating 50, ease of use rating 50, power features 50, product stability 50, enjoyability 50, value 50, product quality importance =*

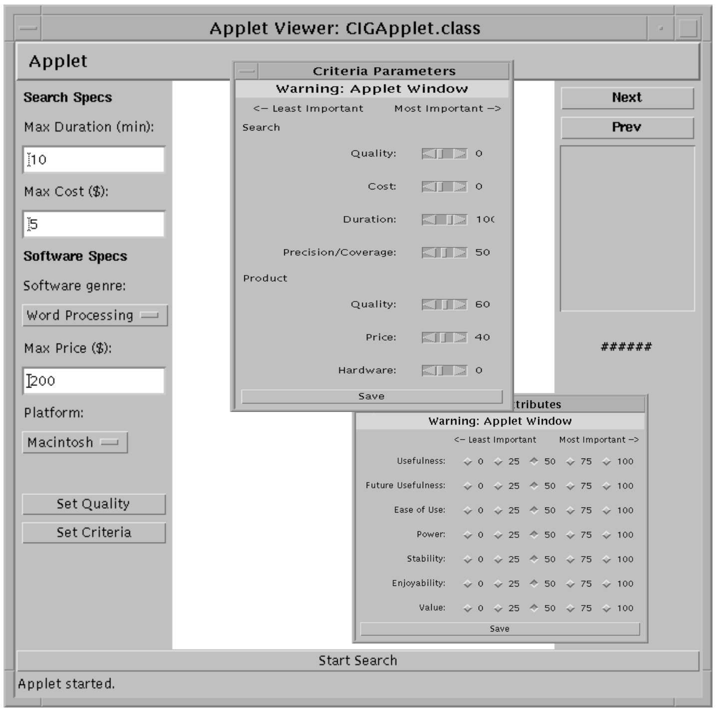

Figure 1: BIG's User Interface

60% and product price importance = 40%. The client is an experienced home-office user who does not want to wait forever for a result and does not want to spend much on the search, and who is primarily concerned with getting most power for the dollar product-wise.

Upon receipt of the criteria, BIG first invokes its planner to determine what information gathering activities are likely to lead to a solution path; candidate activities include retrieving documents from known wordprocessing software makers such as Corel and Microsoft as well as from consumer sites containing software reviews, such as the Benchin Web site. Other activities pertain to document processing options for retrieved text. For a given document, there are a range of processing possibilities each with different costs and different advantages. For example, the heavyweight information extractor pulls data from freeformat text and fills templates and associates certainty factors 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 planner handles the process of laying out the problem solving options by emitting a TÆMS [11, 29] task structure that describes alternative ways to perform tasks and quantifies them statistically via discrete probability distributions in terms of quality, cost, and duration (omitted from the figure for clarity). Figure 2 shows the TÆMS task structure produced in response to the client's query. The top level task is to satisfy the user's query, and it has three subtasks *Get-Information*, *Benchin-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()* also specify the combinations of subtasks that may be employed and also the sequence in which to perform them (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 *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) denoting that *Get-Basic-Information* must produce quality before *Get-Extra* 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 5.3). 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" trade-offs quality for duration and cost. There is another element of choice in BIG, it is in the level 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).

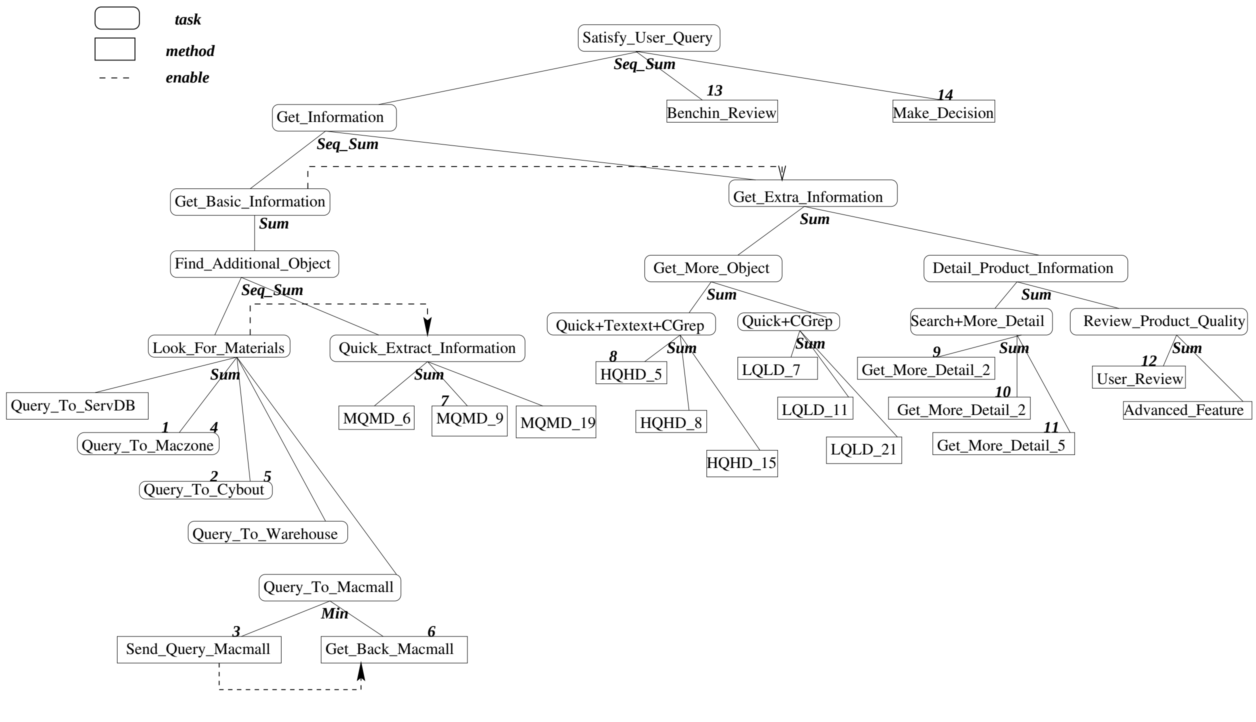

Figure 2: Information Gathering Task Structure

These problem solving options are then considered and weighed by the scheduler – it performs quality/cost/time trade-off analysis and determines a course of action for BIG. The resulting schedule is denoted by the integer annotations on the tasks. The schedule is executed; multiple retrieval requests are issued and documents are retrieved and processed. Data extracted from documents at the Corel site is integrated with data extracted from documents at the Benchin site to form a product description object of Corel WordPerfect. Furthering processing leads to the discovery of 14 other competing products. At the end of the prescribed deadline, BIG's data indicates that the "best" product is Corel WordPerfect 3.5, i.e., it best satisfices the product specifications. BIG returns this recommendation to the client along with the gathered information, corresponding extracted data, and certainty metrics about its extraction and decision processes.

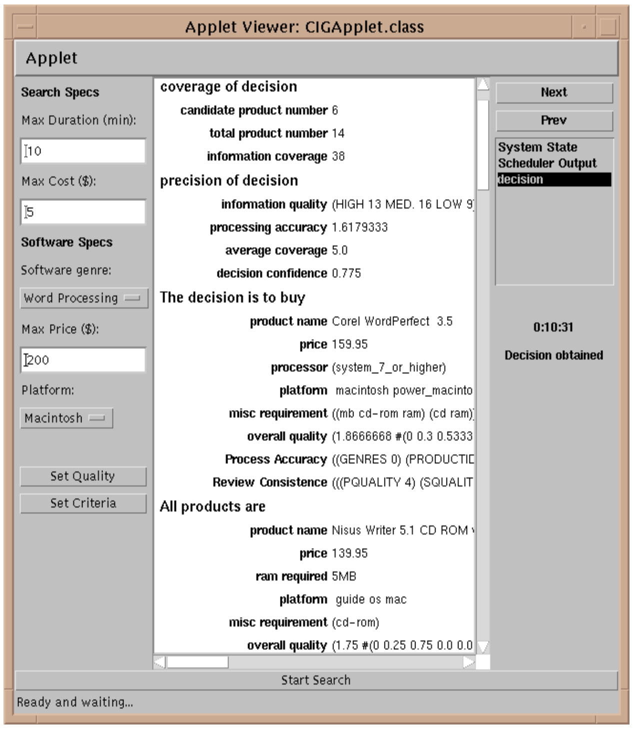

Figure 3: BIG's final decision for this Sample Run

In the remainder of this paper, we discuss related research, Section 2, and then present the BIG agent architecture and its key components in Section 4. In Section 5 we present interesting research issues addressed by BIG and provide details from actual BIG runs. In Section 6 BIG is discussed from a holistic perspective, including empirical experiment results and an execution trace. Conclusions and future directions are presented in Section 7.

2 Related Research

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 [15], is to "move up the information food chain," in other words, to build higher-level information processing engines that utilize existing tools like the URL search engines (e.g., Infoseek and AltaVista). One class of work toward this end is the *meta* search engine. Meta search engines typically issue queries to multiple search engines like AltaVista and Infoseek in parallel, 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 [21] and MetaCrawler [15]; commercial meta search products are also available [22, 42]. 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 features include clustering candidate documents according to similarity and combining redundant URLs. These tools build on the services of the URL search engines, but, the processing done on the retrieved documents is typically limited to the same techniques used to implement the search engines. This often results in wider Internet coverage, but, the output is often just a list of URLs for the human client to process, thus it also suffers from the same problem as the search engines themselves – too much data.

A closely related class of work is the *personal information agent* [1, 35]. Rather than making single queries to a large number of sites, these agents will actively pursue links to find other relevant information. 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 products, 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, the user sacrifices speed for document quality.

Another class of work targeted at moving up the food chain is the shopping agent class. 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 [25] and the more recent Shopbot [14] 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 of the functionality offered by the personal information agents. For example, the commercial Jango [24] shopping agent locates reviews as well as extracting price and very specific product features from vendor web sites. Research aspects of the shopping class often focus on how to autonomously learn (akin to wrappers [34]) to interact with each store's forms and how to learn to process the output, this is in contrast to having a human perform the specification.

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 attempts to push these ideas to the next level.

Like the meta search engines, BIG may 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 the web (in conjunction with web search engines), however, BIG does not stop at locating relevant information, but instead processes it. Like the shopping agents, BIG gathers information to support a decision process. However, BIG differs from the shopping agents in the complexity of its decision process (BIG is not just examining product prices) 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.

BIG is also related to the WARREN [10] multi-agent portfolio management system, which retrieves and process information via 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 is also akin to more database-centric, or structured resource, approaches like TSIMMIS [18], SIMS [2], and the Information Manifold [33], 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 akin to [20, 19, 9, 36] and formal methods [16] 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 focusing on the trade-offs of different information resources, i.e., our work addresses both agent control level and information value.

From the perspective of a digital library, our research is aimed at partially automating the function of a sophisticated research librarian, akin to [41]. 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.

3 Information Gathering as Interpretation to Support a Decision Process

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

The second observation that shapes our solution is that WWW-based IG is an instance of the *interpretation problem*. Interpretation is the process of constructing high-level models (e.g. product descriptions) from low-level data (e.g. raw documents) using feature-extraction methods that can produce evidence that is incomplete (e.g. requested documents are unavailable or product prices are not found) or inconsistent (e.g. different documents provide different prices for the same product). Coming from disparate sources of information of varying quality, these pieces of uncertain evidence must be carefully combined in a well-defined manner to provide support for the interpretation models under consideration.

In recasting IG as an interpretation problem, we face a search problem characterized by a generally combinatorially explosive state space. In the IG task, as in other interpretation problems, it is impossible to perform an exhaustive search to gather information on a particular subject, or even in many cases to determine the total number of instances (e.g. particular word processing programs) of the general subject (e.g. word processing) that is being investigated. Consequently, any solution to this IG problem needs to support reasoning about tradeoffs among resource constraints (e.g. the decision must be made in 1 hour), the quality of the selected item, and the quality of the decision process (e.g. comprehensiveness of search, effectiveness of IE methods usable within specified time limits). Because of the need to conserve time, it is important for an interpretation-based IG system to be able to save and exploit information about pertinent objects learned from earlier forays into the WWW. Additionally, we argue that an IG solution needs to support *constructive problem solving*, in which potential answers (e.g. models of products) to a user's query are incrementally built up from features extracted from raw documents and compared for consistency or suitability against other partially-completed answers.

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

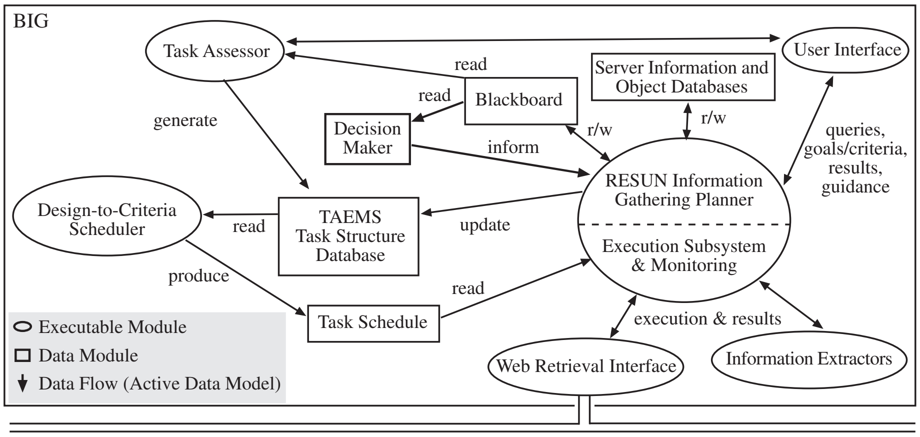

Figure 4: The BIG Agent Architecture

4 The BIG Agent Architecture

The overall BIG agent architecture is shown in Figure 4. The agent is comprised of several sophisticated components that are complex problem 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.

Task Assessor The task assessor is responsible for formulating an initial information gathering plan and then for revising the plan as new information is learned that has significant ramifications for the plan currently being executed. The task assessor is not the execution component nor is it the planner that actually determines the details of how to go about achieving information gathering goals; the task assessor is a component dedicated to managing the high-level view of the information gathering process and balancing the end-to-end top-down approach of the agent scheduler (below) and the opportunistic bottom-up RESUN planner (also below).

Object Database Stores information objects built by BIG during an information gathering session. Objects may be incomplete and uncertainties associated with the fields of the object are explicitly identified. This enables BIG to plan to find information that either corroborates the current information (decreasing the degree of uncertainty) or conflicts with the current information (increasing the degree of uncertainty). The object database is also used to store information from previous searches thus BIG learns and continues to improve and refine its knowledge.

Server Information Database The server 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. Database searches are optimized with the use of both field content indices and an overall database map. As a characterization tool, the database is also able to consolidate or cluster related records in order to obtain an approximate representation of certain classes of sources. The server database is expanded during search (new entries are added as new sources are explored) and supplemented by an off-line indexing web spider.

TÆMS Modeling Framework The TÆMS [11] task modeling language is used to hierarchically model the information gathering process and enumerate alternative ways to accomplish the high-level gathering goals. The task structures probabilistically describe the quality, cost, and duration characteristics of each primitive action and specify both the existence and degree of any interactions between tasks and primitive methods. For instance, if the task of *Find-Competitors-for-WordPerfect* overlaps with the task of *Find-Competitors-for-MS-Word* (particular bindings of the general *Find-Competitors-for-Software-Product* task) then the relationship is described via a mutual facilitation and a degree of the facilitation specified via quality, cost, and duration probability distributions. TÆMS task structures are stored in a common repository and serve as a domain independent medium of exchange for the domain-independent agent control components; in the single agent implementation of BIG, TÆMS is primarily a medium of exchange for the scheduler, below, the task assessor, and the RESUN planner.

Design-to-Criteria Scheduler Design-to-Criteria [39, 40] is a domain independent real-time, flexible computation [19, 9, 36] approach to task scheduling. The Design-to-Criteria task scheduler reasons about quality, cost, duration and uncertainty trade-offs of different courses of action and constructs custom satisficing schedules for achieving the high-level goal(s). The scheduler provides BIG with the ability to reason about the trade-offs of different possible information gathering and processing activities, in light of the client's goal specification (e.g., time limitations), and to select a course of action that best fits the client's needs and the current problem solving context. The scheduler receives the

TÆMS models generated by the task assessor as input, produces a schedule in soft real-time, returns the generated schedule to the RESUN planner for execution. The schedule may contain segments of parallel activities when the primitive actions entail non-local processing, e.g., issuing requests over the network. The integration of the scheduler in the BIG project led to the discovery of an interesting problem with the satisficing methodology used by the scheduler. We discuss this aspect of the work in Section 5.2.

RESUN Planner The RESUN [5, 6] (pronounced "reason") blackboard based planner/problem solver directs information gathering activities. The planner receives an initial action schedule from the scheduler and then handles information gathering and processing activities. The strength of the RESUN planner is that it identifies, tracks, and plans to resolve sources-of-uncertainty (SOUs) associated with blackboard objects, which in this case correspond to gathered information and hypotheses about the information. For example, after processing a software review, the planner may pose the hypothesis that Corel Wordperfect is a Windows98 word processor, but associate a SOU with that hypothesis that identifies the uncertainty associated with the extraction technique used. The planner may then decide to resolve that SOU by using a different extraction technique or finding corroborating evidence elsewhere. RESUN's 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, 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 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 7).

Blackboard Component The Blackboard functions as a multileveled database for the information the system has discovered and produced thus far. Our current blackboard organization has four levels: User-Goal, Decision, Object, and Document, in order of decreasing granularity. The layered hierarchy allows for explicit modeling of concurrent top-down and bottom-up processing, while maintaining a clear evidential path for supporting and contradictory information. The information at a given level is thus derived from the level(s) below it, and it in turn supports the hypotheses at higher levels. For example, when evaluating a particular decision hypothesis' appropriateness, the system would examine the reliability of the text extraction processes used to generate the properties of the object upon which the decision will be performed, each of which are in turn supported by various documents which have been retrieved. An additional level, for an object's features, is also soon to be added, which will facilitate data association and co-referencing problems.

Web Retrieval Interface The retriever tool is the lowest level interface between the problem solving components and the Web. The retriever fills retrieval requests by either gathering the requested URL or by interacting with with both general (e.g., InfoSeek), and site specific, search engines.

Document Classifiers As we discuss in Section 5.1, BIG relies on a set of document classification tools to filter out documents that resemble documents relevant to the query, but are actually *distractions* to BIG if processed and added to its product information base.

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. [2] The information extractors are implemented as knowledge sources in BIG's RESUN planner and are invoked after documents are retrieved and posted to the blackboard. The information extractors are:

textext-ks This knowledge source processes unstructured text documents using the BADGER [38] information extraction system to extract particular desired data. The extraction component uses a combination of learned domain-specific extraction rules, domain knowledge, and knowledge of sentence construction to identify and extract the desired information. This component is a heavy-weight NLP style extractor that processes documents thoroughly and identifies uncertainties associated with extracted data.

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 extraction s 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.

tablext-ks This specialized KS extracts tables from html documents, processes the entries, and fills product description slots with the relevant items. This KS is trained to extract tables and identify table slots for particular sites. For example, it knows how to process the product description tables found at the Benchin review site.

quick-ks This fast and highly specialized KS is trained to identify and extract specific portions of regularly formatted html files. For example, many of the review sites use standard layouts.

[2] The advent of XML and other structuring specifications for web documents will help to simplify the problem of processing web-based information.

Decision Maker After product information objects are constructed BIG moves into the decision making phase. In the future, BIG may determine during decision making that it needs more information, perhaps to resolve a source-of-uncertainty associated with an attribute that is the determining factor in a particular decision, however, currently BIG uses the information at hand to make a decision. We discuss the decision process in greater detail in Section 5.6, 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.

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

5 Interesting Facets of BIG

In this section we present and discuss different facets of BIG from an isolated perspective. More holistic views of BIG are given in Sections 6.2 and 6.1 where a detailed walk through and aggregate empirical results are presented.

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, 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 and so forth. 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.

Figure 5 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 [8] 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 the first run, shown in the figure, no filter and no 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

	# Rejected	Top Candidates	Selected Product
No Filter or Classifier	0 / 13	Portuguese Dictionary Module Norwegian (Nynorsk) Dictionary Module Norwegian (Bokmal) Dictionary Module The Nisus Dictionary Collection US Definition Dictionary	Portuguese Dictionary Module
Simple Filter	31 / 44	EndLink 2.0 Spelling Coach Pro 4.1 Retrieve It! 2.5 Nisus Writer 5.1 CD ROM with Manual Microsoft Word 6.01	EndLink 2.0
Filter & Classifier	53 / 74	ClarisWorks Office 5.0 Corel WordPerfect 3.5 ACADEMIC Nisus Writer 5.1 CD ROM with Manual Corel WordPerfect 3.5	ClarisWorks Office 5.0

Figure 5: Advantages of Document Classification

the target product class, i.e., 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. Almost all top-ranked candidates are word processing products and the top product, "ClarisWorks Office 5.0" is an integrated office suit that includes a word processing package.

Clearly, document pre-classification at the agent side is necessary. Vendor search engines are typically keyword based and generally 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.

5.2 Scheduling for Hard-Deadlines

Design-to-Criteria (DTC) scheduling is the soft real-time process of evaluating the quality, cost, duration, and certainty trade-offs of alternative ways to achieve a task, and producing a custom schedule for task achievement that meets the requirements, e.g., real-time deadlines, cost constraints, quality preferences, etc., of the client. DTC is an approach to coping with exponential combinatorics through satisficing, approximation, and goal-directed schedule generation, all of which is documented in [40].

During the course of the BIG project, we encountered an interesting problem with the satisficing focusing methodology used in Design-to-Criteria when it is combined with hard deadlines and certain classes of very large task structures. Without delving into exhaustive detail, the problem is that in order to cope with the high-order combinatorics in these particular situations, the scheduling algorithm must prune schedule approximations, called *alternatives*, and develop only a subset of these. Herein lies the problem.

Alternatives are constructed bottom-up from the leaves of the task hierarchy to the top-level task node, i.e., the alternatives of a task are combinations of the alternatives for its sub-tasks. Figure 6 shows the alternative set generation process for a small task structure. Alternatives are generated for the interior tasks T_1 and T_2, and these alternatives are combined to produce the alternative set for the root task, T. The complexity of the alternative generation process is pronounced. A task structure with n methods leads to $O(2^n)$ possible alternatives at the root level. We control this combinatorial complexity by focusing

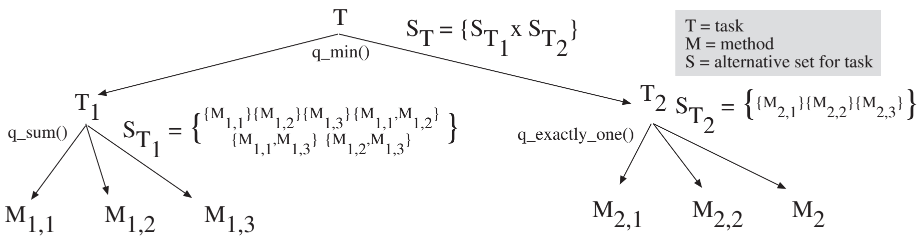

Figure 6: Alternative Sets Lead to Cumbersome Combinatorics

alternative generation and propagation on alternatives that are most likely to result in schedules that "best" satisfice to meet the client's goal criteria; alternatives that are less good at addressing the criteria are pruned from intermediate level alternative sets. For example, a criteria set denoting that certainty about quality is an important issue will result in the pruning of alternatives that have a relatively low degree of quality certainty. After the alternative set for the high-level task is constructed, a subset of the alternatives are selected for scheduling.

For situations in which there is no overall hard deadline, or in which shorter durations are also preferred, the focusing mechanism works as advertised. However, in the BIG project, we are also interested in meeting real-time deadlines and other hard resource constraints (in contrast to those that are relaxable), and often these preferences are not accompanied by a general preference for low duration or low cost. In these cases, the problem lies in making a *local* decision about which alternatives to propagate (at an interior node) when the decision has implications to the local decisions made at other nodes – the local decision processes are interdependent and they interact over a shared resource, e.g., time or money. Casting the discussion in terms of Figure 6: assume T has an overall deadline of 5 minutes and T_1's alternatives require anywhere from 2 minutes to 20 minutes to complete, and T_2's alternatives are similarly characterized. Assume that quality is highly correlated with duration, thus the more time spent problem solving, the better the result. If the criteria specifies maximum quality within the deadline, the alternatives propagated from T_1 to T will be those that achieve maximum quality (and also have high duration). Likewise with the alternatives propagated from T_2. The resulting set of alternatives, S_T at node T will contain members characterized by high quality, but also high duration, and the scheduler will be unable to construct a schedule that meets the hard deadline. The optimal solution to this problem is computationally infeasible ($\omega(2^n)$ and $o(n^n)$) as it amounts to the general scheduling problem because of task interactions and other constraints.

Two approximate solutions are possible. One approach is to preprocess the task structure, producing small alternative sets at each node that characterize the larger alternative population for that node. Then examining the ranges of alternatives at each node and heuristically deciding on an allocation or apportionment of the overall deadline or cost limitation to each of the interior nodes. This local-view of the overall constraint could then be used to focus alternative production on those that will lead to a root-level set that meets the overall constraint. The other approach, which we have employed, is to detect when the local-view of the decision process is problematic and in those cases sample from the population of alternatives, producing a subset that exhibits similar statistical properties, and propagating these alternatives. This leads to a less-focused set of root level alternatives than the prior approach, but it saves on the added polynomial level expense of the first approach; this solution has served us well in the BIG project and enabled the scheduler to maintain its soft real-time level of performance and still produce good schedules.

5.3 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 most constrained resource. 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 .3, this expresses the idea that 30% of any additional time should be spent improving precision and 70% should be spent discovering new products.

#	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.

Table 1: Trading-Off Precision and Coverage

BIG achieves this trade-off behavior in two ways: by planning and scheduling for it 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 for the Design-to-Criteria scheduler and giving the scheduler a set of options, from which to choose a course of action. In Figure 2, *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* 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 the value of these, 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.

As mentioned, BIG can also work opportunistically to improve coverage or precision, at its description. A third option, not currently implemented, is for BIG to revise its problem solving options 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, certainty trade-offs of its options from a unified perspective.

Table 1 shows BIG's ability to trade-off precision and coverage. 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 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 Figures 7(a) and 7(b).[3] 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 7(a) begins a series of *Median_Quality_Duration* and *Low_Quality_Duration* activities that retrieve and process additional product related documents. The postfixed integers on the method names, e.g.,six, eight, six, eight, and seven, respectively, denote the number of documents 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 7(b) begins a series of *Get_More_Detail* actions that seek to find information about existing product objects.

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

[3]We will provide drawn schedules annotated with statistical information for the final version of this paper.

Start Time (as 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_Median_Quality_Duration_6
93	method9_Median_Quality_Duration_8
165	method21_Median_Quality_Duration_6
221	method20_Median_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

(a) Schedule for 10% / 90% Precision to Coverage

Start Time (as 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_Median_Quality_Duration_8
111	method21_Median_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

(b) Schedule for 90% / 10% Precision to Coverage

Figure 7: Different Precision/Coverage Schedules

5.4 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 maker for the product. When BIG encounters this fusion issue, the item with the highest associated degree of belief is used. Another issue is how to integrate different opinions about the product. The latter is done in BIG by associating two metrics with every 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 Figure 8, 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.[4] The result of this extraction is the partial product object shown in Figure 9(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

[4]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.

```
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/dramatica10.htm
URL_E  http://www.zdnet.com/macuser/mu_0796/reviews/review12.html
URL_F  http://www.macaddict.com/issues/0797/rev.dramaticapro.html
```

Figure 8: URLs for Documents Retrieved During Processing

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 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 Windows95 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, the it is given a rating of -.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 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 (squality) rating and the negative review (pquality) rating will thus have little weight when compared to other sources. The product object after this step is shown in Figure 9(b).

```
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:  2
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

Figure 9: Evolution of the Dramatica Product Object

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. Figure 9(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 Figure 9(d). The final product object is subsequently compared to other product objects during the decision process (Section 5.6. 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.

5.5 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 of 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 the 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 `http://macworld.zdnet.com/pages/april.97/Reviews.3334.html`. 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 entail 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 with for the *Mac_Publishing* and *WordPerfect* objects. The choice is determined by the precision versus coverage specification (Section 5.3) 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 of the new products rather than follow the original schedule. Method *Get_More_Detail_1* is executed and the CyberianOutpost 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 the following new information about the product:

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

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

5.6 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. Since there are several features for one product, such as “price,” “quality,” “hardware,” the score represents each feature based on how important it is to the client. 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 an information source is based on human knowledge and prior experience about this source. Unknown sources are ranked based on their reference number – the more times a source is referenced by other URLs, the higher quality it may be. 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 are constructed from expert advice, prior human examination and learning. These tables provide more accurate descriptions of those information sources.

For example, for the product “Corel WordPerfect,” review from site A says it is very good: Product Quality(PQUALITY) = 4, review from site B says it is good : PQUALITY=3. Site A is known to be a medium quality site: Source Quality(SQUALITY = 2), the quality measure table of site A is shown in Figure 10. Site B is a high quality site, the quality measure table of site B is shown in Figure 11.

Review Quality	P robability of achieving true quality given in Row 0				
	5	4	3	2	1
5	0.1	0.2	0.5	0.1	0.1
4	0.0	0.3	0.2	0.3	0.2
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

Figure 10: Review Quality for Site A

Review Quality	P robability of achieving true quality given in Row 0				
	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
3	0.0	0.2	0.7	0.1	0.0
2	0.0	0.0	0.2	0.7	0.1
1	0.0	0.0	0.0	0.3	0.7

Figure 11: Review Quality for Site B

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, 35% 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 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 choose decision coverage and precision as two main characteristics of the decision.

Decision Coverage is a 3-dimension vector:

1. **Total Product Number** Indicates how many products the agent has found; the more products the agent finds, the higher quality of the decision is.
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** Indicates the number of documents the agent has processed.

Decision Precision is a 4-dimension 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 extracting process is not perfect for any document, the information extractor gives the degree of belief for every item it returns. For example, textext-ks may say it finds the operating system for "Corel WordPerfect" is "mac," the degree of belief is 0.8. This number comes from the information extractor knowledge source. The information 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 score distribution of the discovered products. For example, if product A has score distribution(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;

Decision evaluation gives the client a more critical view of the decision.

6 BIG in Action

In this section we present and discuss different BIG from a holistic perspective.

6.1 Empirical Results

Table 2 illustrates how the system operations under different time constraints. The experiments are for searches to find word processing products and the search and product criteria is the same for all runs. Only the time alloted for the search varies.

The first four columns of data provide information about the duration of each search. *User Time* denotes the users target search time; the value in parenthesis 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 tradeoff. (Utility in these cases is linearly decreasing between the expressed deadline and 10% above the expressed deadline.)[5] *Schedule* 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, slow downs 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 (#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

[5]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.

User Time	#	Scheduled	Execution	#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 = users 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, #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
.

Table 2: Different Time Allotments Produce Different Results

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, 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 is not dependent on execution time or processes effort.

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 contract 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.

6.2 Execution Trace

We now describe a short sample run of the BIG system. 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. The run is split into the following processes: querying, planning, scheduling, retrieval, extraction and decision making.

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% 40% respectively. Product quality is viewed as a multi-dimensional attribute with features like usefulness, future usefulness, stability, value, ease of use, power and enjoyability constituting the different dimensions. These are assigned relative weights of importance. The client specifies the relative importance of product coverage and precision as 20% and 80% respectively.

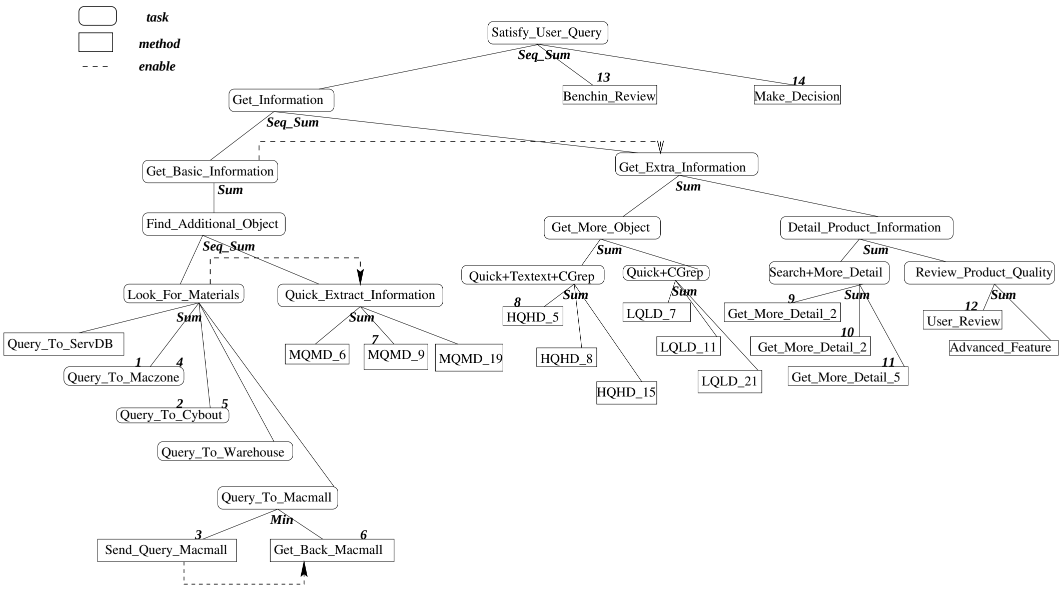

Figure 12: BIG's TÆMS Task Structure for the Short Run

Once the query is specified, the task assessor starts the process of analyzing the client specifications. Then, 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 is akin to a process plan for achieving a particular task). 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. The task structure produced for our sample query is shown in Figure 12.

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 Figure 12. The numbers near particular methods indicate their assigned execution order. The scheduled time and execution time of each method are shown in the following table:

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
Median_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 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), a computer retailer. While this information is being retrieved, a second query is made to another retailer site, the Cyberian Outpost (www.cybout.com), a third query is made to MacMall(www.cc-inc.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 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 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 to the blackboard. Out of these candidate document descriptions, 13 documents are chosen for MedimumQuality(MQMD)_9 processing. The choice is made heuristically and is based on recency of information in the document, length of the document and its source site (some sites are preferred over others).

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, the other product is a speech recognition software . 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. An abbreviated version of the object follows: [6] After quickext-ks, the object is:

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

The cgrep-ks finds extra information about the products processor, platform, and miscelaneous requirements. It also finds corroborating product name and price information; this increases the processing accuracy 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(Degree of Belief):
    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 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 Macintosh platform are discarded. Subsequent efforts are focused on the remaining six products.

The following three methods *Get_Detail* make queries to "yahoo" and "infoseek" about the remaining products and find some review documents. A review process ks 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 pictured below. Product Quality denotes the quality of the product as extracted from the review (in light of the client's goal criteria), and Search Quality 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 `SQUALITY` is determined based on the reference number of the document, the more widely a document is referenced, the more highly it is rated.

- `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/`

[6] Ironically, given the semi-structured nature of the bullet list in the source document that identifies the hard disk and ram requirements, the extraction methods failed to properly extract these requirements. A specialized information extractor for data in this particular format (nested bullet lists) is one way to properly handle this extraction case.

For example, four documents (above) are found and processed as review documents for product "Corel WordPerfect 3.5." After processing, 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(Degree of Belief):
    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)))
```

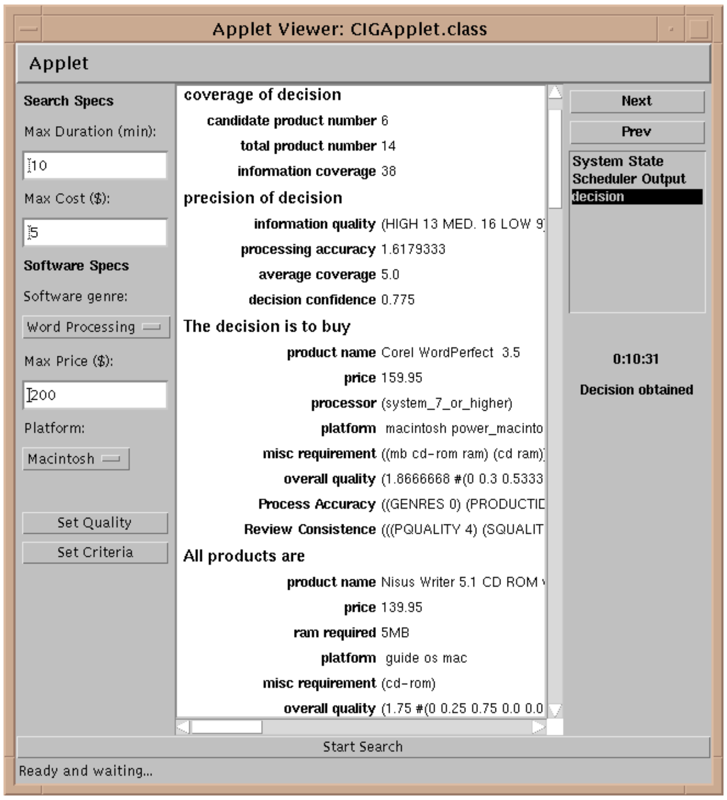

Figure 13: BIG's final decision for this Sample Run

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 of general search engine and natual language process technology. Thus it is necessary to get information for some specific product review site. 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`, which includes 60 users' reviews, is selected and processed for product “Microsoft Word 6.01”, Word 6.01 being one of the six competing products still in the fray. The new review information `((PQUALITY 2.857143) (SQUALITY 3))` is added to “Microsoft Word 6.01” object. Similarly, `Benchin-Review-Method` sends queries to the Benchin star review site (uses a star rating system that is simple to process), producing review information for four different products.

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. Detailed decision-making process is described Section 5.3. The final decision is shown in Figure 13. The decision confidence is not very high because there is a competing candidates “Nisus Writer 5.1 CD ROM with Manual” whose overall quality is slightly less than that of ”Corel WordPerfect”.

7 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.

Another feature of BIG not stressed 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.

Also not stressed in this paper is BIG’s ability to learn from prior problem solving instances [30]. 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 and adapting BIG to operate in another domain, perhaps the auto-purchase domain, would require the addition of specific knowledge about the particular domain. For example, information extractors like the information extraction system, cgrepext-ks, and quickext-ks, require supervised training to learn extraction rules and make use of semantic dictionaries to guarantee a certain level of performance. (Though we tested the system on the related domain of computer hardware and found it to work well considering no hardware related documents were in the training corpus.) Additionally, both the server and object databases, being persistent stores of the system's past experiences, are inherently domain dependent, rendering most of this knowledge useless and possibly distractive when used in other scenarios.

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 non-trivial 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 future direction involves moving BIG into a multi-agent system. Our group [23] has a long history of developing distributed problem solving and multi-agent systems [12, 13, 26, 37, 6, 32] and we are interested in exploring multi-agent coordination via a group of agents descended from the current BIG agent.

8 Acknowledgments

We would like to thank Professor Norman Carver for his contributions relating to the RESUN planner and his input on the task assessor component. We would also like to acknowledge the help of Mike Chia during the formative stages of this project.

References

[1] Autonomy agentware technology white paper. http://www.agentware.com/main/tech/whitepaper.htm.

[2] Y. Arens, C. Knoblock, and W. Shen. Query reformulation for dynamic information integration. *Journal of Intelligent Information Systems, Special Issue on Intelligent Information Integration*, 6(2 & 3):99–130, 1996.

[3] C. Mic Bowman, Peter B. Danzig, Udi Manber, and Michael F. Schwartz. Scalable Internet Resource Discovery: Research Problems and Approaches. *Communications of the ACM*, 37(8):98–114, 1994.

[4] J. P. Callan, W. Bruce Croft, and S. M. Harding. The INQUERY retrieval system. In *Proceedings of the 3rd International Conference on Database and Expert Systems Applications*, pages 78–83, 1992.

[5] Norman Carver and Victor Lesser. A new framework for sensor interpretation: Planning to resolve sources of uncertainty. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 724–731, August 1991.

[6] Norman Carver and Victor Lesser. The DRESUN testbed for research in FA/C distributed situation assessment: Extensions to the model of external evidence. In *Proceedings of the First International Conference on Multiagent Systems*, June, 1995.

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

[8] Naive bayes document classifier. Supplement to Machine Learning by Tom Mitchell, 1997, McGraw Hill. http://www.cs.cmu.edu/afs/cs/project/theo-11/www/naive-bayes.html.

[9] T. Dean and M. Boddy. An analysis of time-dependent planning. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 49–54, St. Paul, Minnesota, August 1988.

[10] K. Decker, A. Pannu, K. Sycara, and M. Williamson. Designing behaviors for information agents. In *Proceedings of the 1st Intl. Conf. on Autonomous Agents*, pages 404–413, Marina del Rey, February 1997.

[11] Keith S. Decker. Task environment centered simulation. In M. Prietula, K. Carley, and L. Gasser, editors, *Simulating Organizations: Computational Models of Institutions and Groups*. AAAI Press/MIT Press, 1996. Forthcoming.

[12] Keith S. Decker and Victor R. Lesser. Designing a family of coordination algorithms. In *Proceedings of the First International Conference on Multi-Agent Systems*, pages 73–80, San Francisco, June 1995. AAAI Press. Longer version available as UMass CS-TR 94–14.

[13] K.S. Decker, V.R. Lesser, M.V. Nagendra Prasad, and T. Wagner. MACRON: an architecture for multi-agent cooperative information gathering. In *Proceedings of the CIKM-95 Workshop on Intelligent Information Agents*, Baltimore, MD, 1995.

[14] Robert Doorenbos, Oren Etzioni, and Daniel Weld. A scalable comparision-shopping agent for the world-wide-web. In *Proceedings of the First International Conference on Autonomous Agents*, pages 39–48, Marina del Rey, California, February 1997.

[15] Oren Etzioni. Moving up the information food chain: Employing softbots on the world wide web. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 1322–1326, Portland, OR, August 1996.

[16] Oren Etzioni, Steve Hanks, Tao Jiang, Richard Karp, Omid Madani, and Orli Waarts. Optimal information gathering on the internet with time and cost constraints. In *Proceedings of the Thirty-seventh IEEE Symposium on Foundations of Computer Science (FOCS)*, Burlington, VT, October 1996.

[17] D. Fisher, S. Soderland, J. McCarthy, F. Feng, and W. Lehnert. Description of the UMass Systems as Used for MUC-6. In *Proceedings of the 6th Message Understanding Conference*, Columbia, MD, 1996.

[18] J. Hammer, H. Garcia-Molina, K. Ireland, Y. Papakonstantiou, J. Ullman, and J. Widom. Information Translation, Mediation, and Mosaic-based Browsing in the TSIMMIS System. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 1995.

[19] Eric Horvitz, Gregory Cooper, and David Heckerman. Reflection and action under scarce resources: Theoretical principles and empirical study. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, August 1989.

[20] Eric Horvitz and Jed Lengyel. Flexible Rendering of 3D Graphics Under Varying Resources: Issues and Directions. In *Proceedings of the AAAI Symposium on Flexible Computation in Intelligent Systems*, Cambridge, Massachusetts, November 1996.

[21] Adele Howe and Daniel Dreilinger. A Meta-Search Engine that Learns Which Engines to Query. *AI Magazine*, 18(2), 1997.

[22] Inforia quest. http://www.inforia.com/quest/iq.htm.

[23] The multi-agent systems laboratory at the university of massachusetts amherst. dis.cs.umass.edu.

[24] Jango. http://www.jango.com/.

[25] Bruce Krulwich. The BargainFinder Agent: Comparison price shopping on the Internet. In Joseph Williams, editor, *Bots and Other Internet Beasties*. SAMS.NET, 1996. http://bf.cstar.ac.com/bf/.

[26] S. Lander and V.R. Lesser. Negotiated search: Organizing cooperative search among heterogeneous expert agents. In *Proceedings of the Fifth International Symposium on Artificial Intelligence Applications in Manufacturing and Robotics*, December 1992.

[27] Leah Larkey and W. Bruce Croft. Combining classifiers in text categorization. In *Proceedings of the 19th International Conference on Research and Development in Information Retrieval (SIGIR '96)*, pages 289–297, Zurich, Switzerland, 1996.

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

[29] Victor Lesser, Keith Decker, Norman Carver, Alan Garvey, Daniel Neiman, Nagendra Prasad, and Thomas Wagner. Evolution of the GPGP Domain-Independent Coordination Framework. Computer Science Technical Report TR-98-05, University of Massachusetts at Amherst, January 1998.

[30] Victor Lesser, Bryan Horling, Frank Klassner, Anita Raja, Thomas Wagner, and Shelley XQ. Zhang. BIG: A resource-bounded information gathering agent. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, July 1998. See also UMass CS Technical Reports 98-03 and 97-34.

[31] Victor Lesser, Bryan Horling, Frank Klassner, Anita Raja, Thomas Wagner, and Shelley XQ. Zhang. A next generation information gathering agent. In *Proceedings of the Fourth International Conference on Information Systems, Analysis, and Synthesis*, pages 41–50, July 1998. In conjunction with the World Multiconference on Systemics, Cybernetics, and Informatics (SCI'98), Volume 2. Also available as UMASS CS TR 1998-72.

[32] Victor 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):89–111, 1998.

[33] Alon Levy, Anand Rajaraman, and Joann J. Ordille. Query-answering algorithms for information agents. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 40–47, Portland, OR, August 1996.

[34] Ion Muslea, Steve Minton, and Craig Knoblock. STALKER: Learning Extration Rules for Semistructured, Web-based Information Sources. In *Proceedings of the AAAI Workshop on AI and Information Integration*, 1998.

[35] Robo surfer. http://www.robosurfer.com/.

[36] Stuart J. Russell and Shlomo Zilberstein. Composing real-time systems. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, pages 212–217, Sydney, Australia, August 1991.

[37] Tuomas Sandholm. An implementation of the contract net protocol based on marginal cost calculations. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 256–262, Washington, July 1993.

[38] S. Soderland, D. Fisher, J Aseltine, and W.G. Lehnert. Crystal: Inducing a conceptual dictionary. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 1314–1321, 1995.

[39] Thomas Wagner, Alan Garvey, and Victor Lesser. Complex Goal Criteria and Its Application in Design-to-Criteria Scheduling. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, pages 294–301, July 1997. Also available as UMASS CS TR-1997-10.

[40] Thomas Wagner, Alan Garvey, and Victor Lesser. Criteria-Directed Heuristic Task Scheduling. *International Journal of Approximate Reasoning, Special Issue on Scheduling*, 19(1-2):91–118, 1998. A version also available as UMASS CS TR-97-59.

[41] M.P. Wellmen, E.H. Durfee, and W.P. Birmingham. The digital library as community of information agents. *IEEE Expert*, June 1996.

[42] Zurfrider. http://www.zurf.com/.