Sophisticated Information Gathering in a Marketplace of Information Providers[1]

V. Lesser, B. Horling, A. Raja, T. Wagner, and X. Zhang

Computer Science Department
University of Massachusetts at Amherst

lesser@cs.umass.edu

Abstract

BIG is a sophisticated, web-based, information gathering agent that recommends software packages. BIG plans, locates and processes free-format WWW documents via natural language processing and other text extraction techniques. BIG uses the processed information to create models of software products and then compares the models to the client's criteria and recommends a product for the client to purchase. BIG also uses the information extracted during the search to adapt and refine the search process itself – for example, deciding to gather more information on a particularly highly referred product. This paper discusses techniques used by BIG to control: 1) the amount of monies spent in acquiring information from sites that charge a fee for accessing their information, 2) the balance between the scope/coverage of information gathered and the precision of resulting decision, and 3) the end-to-end time that the information gathering and processing activities will take. As part of this discussion, we present the DTC (Design-to-Criteria) scheduler and the domain-independent activity representation, called TAEMS, that is used to describe and quantify BIG's problem solving activities for the scheduler. We also present experimental results showing how BIG reconfigures its activities to meet resource and cost constraints.

[1] 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 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.

1. Introduction

Over the last five years, we have been developing a next generation, web-based information gathering (IG) agent, called BIG (resource Bounded Information Gathering). BIG locates, retrieves and processes information to support a human decision process. Specifically, BIG helps clients pick software packages, although the techniques it uses are general enough to apply to a wide range of domains. For example, a client can instruct BIG to recommend a database package for Windows 98 and specify constraints on both the search process (e.g., how much money to spend gathering information) and on the product, e.g., the amount of money he or she is willing to pay for the product, reliability, ease-of-use, etc. BIG will then formulate a plan, locate, and process relevant information, and return a recommendation to the client, along with the supporting data.

Like meta-search engines [15, 11, 16, 33] 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. Like a personal information agent [1, 25], BIG gathers documents by actively searching the web (both by following links and using search engines), however, BIG does not stop at just locating relevant information, but instead can analyze the documents found using a variety of techniques that include a heavy-weight natural language text extraction system, BADGER [27], a variety of lighter weight extraction approaches such as a Unix-style grep and site-specific wrapper utilities. Conceptually, BIG "reads" free-format text, identifies product features like prices, disk requirements, support policies, etc., extracts these features from the documents and then reasons about them. Like shopping agents [19, 8, 17], BIG gathers information to support a decision process. However, BIG differs from a shopping agent in the complexity of its decision process (BIG is not just comparing product prices) and in the complexity of its runtime information processing facilities. BIG is related to the WARREN [5] multi-agent portfolio management system, which also retrieves and processes information, 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. Other related work can also be seen in [12, 28, 14, 32, 26]. An example of a typical client query to BIG and its output is shown in Figure 1.

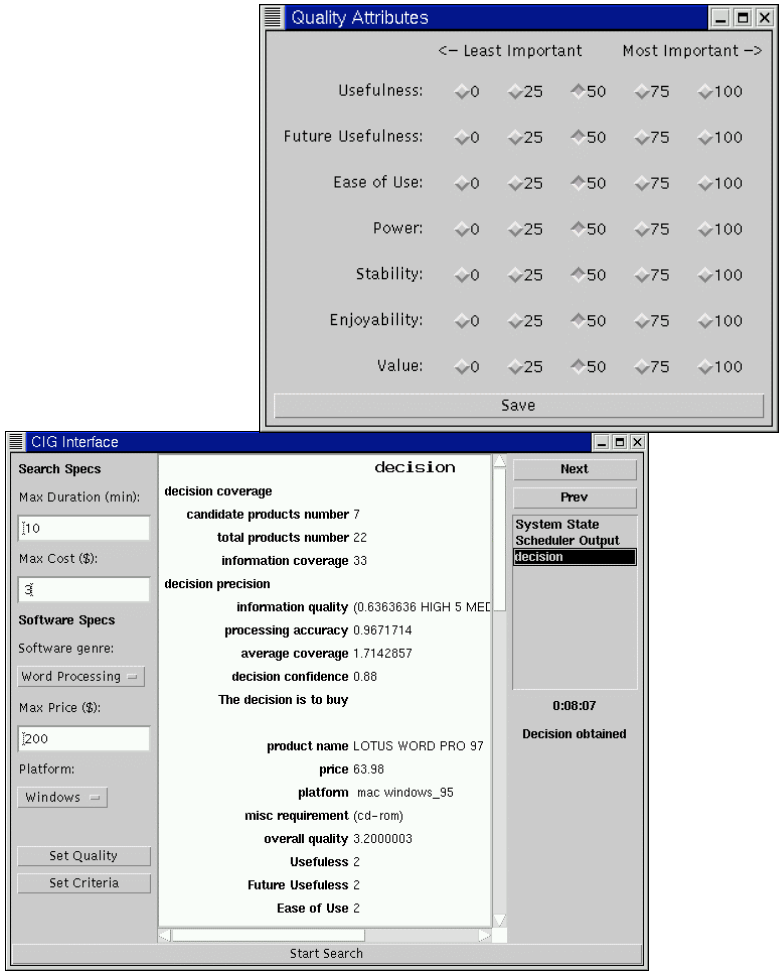

Figure 1: A Query/Response Search Episode

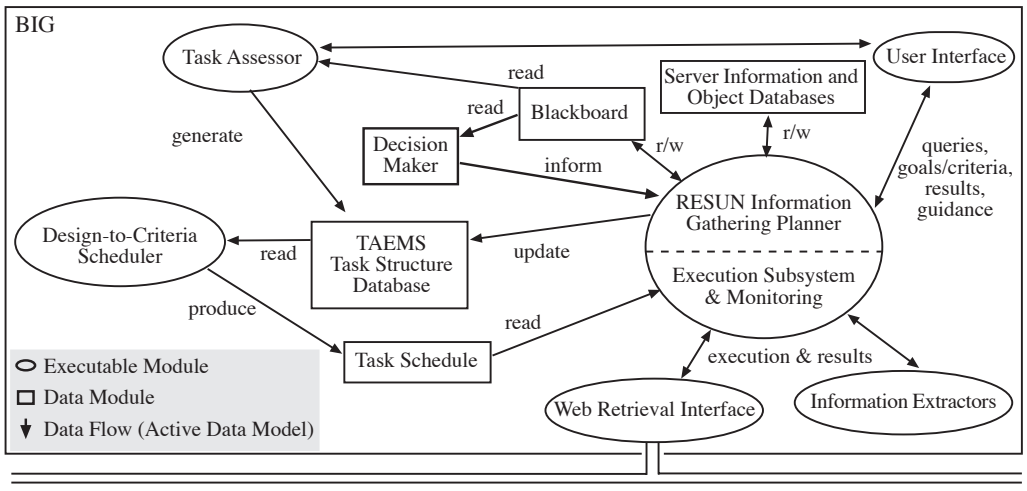

Figure 2: The BIG Agent Architecture

The BIG agent architecture (shown in Figure 2) and its overall performance capabilities have been well documented previously [21, 22, 23]. The following is a brief description of the major components:

RESUN Planner – A blackboard-based interpretation planner that is the domain expert. Gathers and processes information, builds models of products, and recommends purchase decisions.

Information Extractors - Text extraction tools that process free-format text to produce structured data.

Document Classifiers - Qualify text before sending onto the information extractors. Prevents BIG from being distracted by information unrelated to the query at hand.

Server Information Database - While BIG uses commercial index-based engines like AltaVista and Infoseek, BIG also maintains a local database of information sources.

Object Database - Where BIG stores information about products as it constructs models of the products. Also where BIG keeps information learned from prior searches.

Design-to-Criteria (DTC) Scheduler - Agent control problem solver, enables BIG to meet deadlines and cost restrictions.

TAEMS Modeling Framework - Used to quantify and model the agent's problem solving behaviors for the DTC scheduler.

Task Assessor - Responsible for managing the interface between the RESUN opportunistic planner and the DTC scheduler.

In this paper, we focus on a recently developed capability of BIG's: its ability to schedule its information gathering activity in order to control the amount of monies spent in acquiring information from sites that charge a fee for accessing their information. We feel that the model of free access for information on the WWW based on advertisers' fees will, in time, not be the dominant model for WWW. A number of sites, for example Consumer Reports (www.consumerreports.com) and the Wall Street Journal (www.wsj.com), already charge access fees for their information. In turn, they provide high quality, screened information that would be useful for an intelligent assistant. This is especially true as the type of intelligent agent assistance exemplified in systems like BIG become the more common model for how users make purchasing decisions and other high-level decisions which require extensive information gathering on the WWW. Recent work on digital libraries [10] supports this view that there will be an information economy on the WWW. Though our work reported here does not emphasize a marketplace model for acquisition of information where costs can be negotiated, our basic approach will support this more dynamic costing model.

This new capability fits very nicely with BIG's existing ability to control both the end-to-end time that the information gathering will take and the balance between

the scope/coverage of information gathered and the precision of resulting decision. From the perspective of the software domain, this latter point refers to the number of products that are discovered and analyzed versus the amount of detailed and overlapping information that is gathered on each product which is used to make the final decision. More generally, BIG is designed with the view that information gathering on WWW must of necessity be resource-bounded and in different situations, the criterion for how to control the resource-boundedness of the system will change. We feel 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 trade-offs 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 information extraction methods usable within specified time limits). Because of the need to conserve time, it is important for an 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 Section 2, we discuss the general framework that allows BIG to make resource-bounded decisions about how to organize its processing to achieve cost objectives, end-to-end completion time, and scope/precision trade-offs given user specified criteria. Section 4 provides detailed experimental data indicating how BIG reorganizes its search process to live within specific monetary constraints on how much can be spent on accessing information useful in making a software purchasing decision. In Section 5, we conclude the article with a review of the major points.

2. Planning to Address Resource Bounds

BIG addresses time and cost limitations by using the Design-to-Criteria (DTC) domain-independent agent scheduler [30, 31]. The DTC scheduler's expertise lies in analyzing an agent's candidate set of problem solving actions and choosing a course of action for the agent that meets complex, multi-dimensional design criteria. In BIG, the design criteria specifies the relative importance of solution completeness and coverage, as well as cost and time restrictions. Design-to-Criteria achieves domain independence through the TAEMS task modeling framework, which describes and quantifies key activities and key decision points in the agent's problem solving process. Design-to-Criteria interfaces with the IG expert planner, a RESUN interpretation planner [2, 3, 4], through TAEMS and another component called the task assessor that abstracts RESUN's information gathering process into TAEMS models for evaluation and scheduling. It is important to note that the DTC scheduling problem has comm s with both classic scheduling and planning

problems. One aspect of scheduling a TAEMS task structure is deciding which alternative tasks to perform or which methods to use to achieve a particular objective; the other aspect is in determining the best sequence in which to perform the activities. We not only order the activities, we can concurrently schedule non-local activities: e.g. new document retrieval requests to web search engines can be scheduled simultaneously with the local processing of documents already retrieved. This point is particularly important as the element of choice in BIG's problem solving process, i.e., having alternative ways to achieve goals, is what gives DTC the room to maneuver and address resource limitations. Without this element of choice, this flexibility, BIG's problem solving behavior would not be adjustable to different resource situations. The other important intellectual point is that quantification of the different problem solving options is necessary for DTC to make "good" or reasoned choices when comparing the different options. Without quantified choice in BIG's process, BIG's behaviors would be very limited. To truly illustrate the issue of flexibility and choice, and their role in resource bounded information gathering, we must take a closer look at TAEMS and BIG's information gathering process.

TAEMS (Task Analysis, Environment Modeling, and Simulation) is a domain independent task modeling framework used to describe and reason about complex problem solving processes. TAEMS models are used in multi-agent coordination research [5, 31] and are being used in many other research projects, including: cooperative-information-gathering [23], collaborative distributed design [7], intelligent environments [20], coordination of software process [9], and others [4, 29, 13]. Typically a problem solver represents domain problem solving actions in TAEMS, possibly at some level of abstraction, and then passes the TAEMS models on to agent control problem solvers like the Design-to-Criteria scheduler.

TAEMS models are hierarchical abstractions of problem solving processes that describe alternative ways of accomplishing a desired goal; they represent major tasks and major decision points, interactions between tasks, and resource constraints but they do not describe the intimate details of each primitive action. All primitive actions in TAEMS, called methods, are statistically characterized via discrete probability distributions in three dimensions: quality, cost and duration. Quality is a deliberately abstract domain-independent concept that describes the contribution of a particular action to overall problem solving. Duration describes the amount of time that the action modeled by the method will take to execute, and cost describes the financial or opportunity cost inherent in performing the action. Uncertainty in each of these dimensions is implicit in the performance characterization - thus agents can reason about the certainty of particular actions as well as their quality, cost, and duration trade-offs. The uncertainty representation is also applied to task interactions like enablement, facilitation and hindering effects[2], e.g., "10% of the time facilitation will increase the quality by 5% and 90% of the time it will increase

[2] Facilitation and hindering task interactions model soft relationships in which a result produced by some task may be beneficial or harmful to another task. In the case of facilitation, the existence of the result, and the activation of this task interaction, generally increases the quality of the recipient task or reduces its cost or duration.

the quality by 8%." The quantification of methods and interactions in TAEMS is not regarded as a perfect science. Task structure programmers or problem solver generators estimate the performance characteristics of primitive actions. These estimates can be refined over time through learning [3 4] and reasoning components typically replan and reschedule when unexpected events occur.

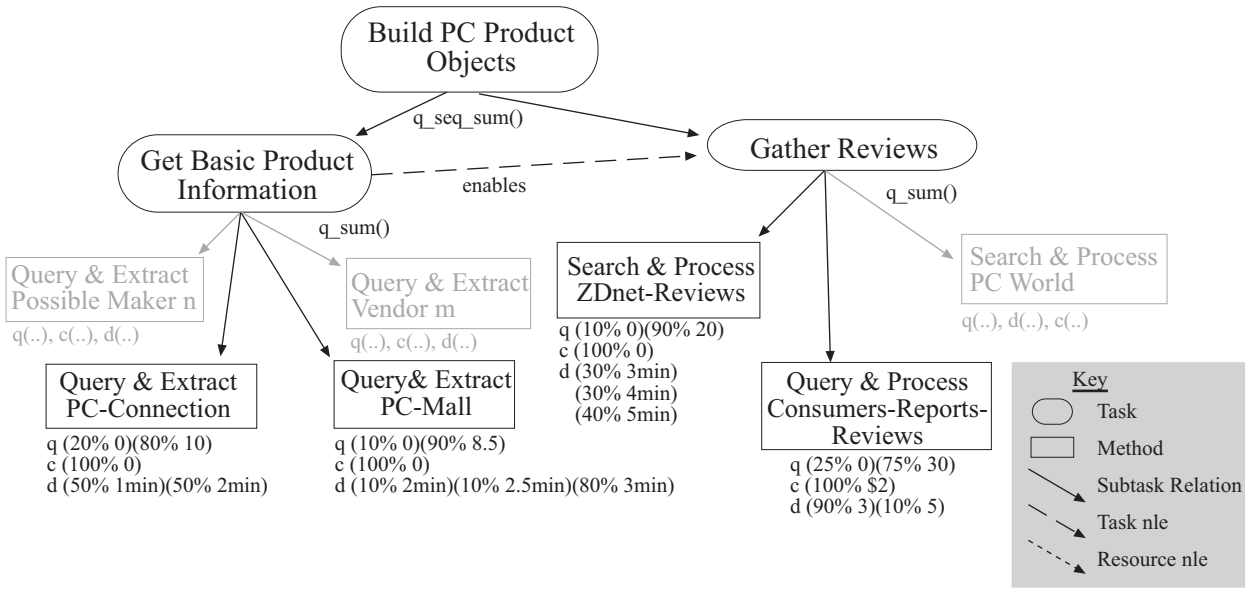

Figure 3: TAEMS IG Task Structure

To illustrate, consider Figure 3, which is a conceptual, simplified sub-graph of a task structure emitted by BIG. The actual task structures are too complex for introductory example purposes - we present an actual TAEMS task structure emitted by BIG in Figure 6. Figure 3 describes the portion of the information gathering process that pertains to constructing models or objects of commercial products. The top-level task is to construct product models of retail PC systems. It has two subtasks, *Get-Basic* and *Gather-Reviews*, both of which are decomposed into methods that are described in terms of their expected quality, cost, and duration. The *enables* arc between *Get-Basic* and *Gather-Reviews* is a non-local-effect (NLE), or task interaction; it models the fact that the review gathering methods need the names of products as a precondition to gathering reviews for them. As we shall illustrate, task interactions are important in BIG's behavior model because they describe the relationship between obtaining more documents and text processing, and solution attributes, such as coverage or precision. Task interactions are also important when we contemplate a multi-agent information gathering scenario because they identify instances in which tasks assigned to different agents are interdependent - they model, in effect, implicit joint goals or joint problem solving activity [2 4]. Coordination is motivated by the existence of these interactions [6].

Returning to the example, *Get-Basic* has two methods, joined under the *sum()* quality-accumulation-function (QAF), which defines how performing the subtasks relate to performing the parent task. In this case, either method or both may be

employed to achieve *Get-Basic*. The same is true for *Gather-Reviews*. The QAF for *Build-PC-Product-Objects* is a *seq_sum()* which indicates that the two subtasks must be performed, in order, and that their resultant qualities are summed to determine the quality of the parent task; thus there are nine alternative ways to achieve the top-level goal in this particular sub-structure. In general, a TAEMS task structure represents a family of plans, rather than a single plan, where the different paths through the network exhibit different statistical characteristics or trade-offs.

TAEMS also supports modeling of tasks that arrive at particular points in time, individual deadlines on tasks, earliest start times for tasks, and non-local methods (those belonging to other agents). In the development of TAEMS there has been a constant tension between representational power and the combinatorics inherent in working with the structure. The result is a model that is non-trivial to process and schedule in any optimal sense (in the general case), but also one that lends itself to flexible and approximate processing strategies.

Given the process described in Figure 3, Design-to-Criteria can construct custom schedules for BIG to meet its current situation. Even this simple task structure gives DTC room to adapt BIG's problem solving. Figure 4 shows four different schedules constructed for different BIG clients that have different objectives and criteria. Schedule A is constructed for a client that has both time and financial resources - he or she is simply interested in maximizing overall solution quality. Schedule B is constructed for a client that wants a free solution. Schedule C is constructed for a client interested in trading-off quality, duration, and cost equally. Schedule D meets the needs of a client interested in maximizing quality while meeting a hard deadline of 7 minutes. Note that schedule D is actually preferred over a schedule that includes method *Query-and-Extract-PC-Connection* even though said method has a higher expected value than *Query-and-Extract-PC-Mall*. This is because the *PC-Connection* method has a higher probability of failure. Because of the *enables* nle from the task of getting product information to retrieving reviews, this higher probability of failure also impacts the probability of being able to query the Consumer's site for a review. Thus, though the local choice would be to prefer *PC-Connection* over *PC-Mall* for this criteria, the aggregate effects lead to a different decision. The general approach to specifying design criteria is a slider metaphor (Figure 5) via which scheduler clients specify the relative importance of quality, cost, duration, certainty in these dimensions, and limits or thresholds on these values.

Schedule A - Client has no resource limitations, maximize quality.

```
        ------------------------------------------------------------------------------------------------------------------------------------
        | Query-and-Extract-PC-Connection | Query-and-Extract-PC-Mall | Query-and-Process-ZDnet | Query-and-Process-Consumers-Reports |
        ------------------------------------------------------------------------------------------------------------------------------------

Quality distribution: (0.02 0.00)(0.00 8.50)(0.00 10.00)(0.02 18.50)(0.04 28.50)(0.02 30.00)(0.18 38.50)(0.01 40.00)
                      (0.05 48.50)(0.12 58.50)  (0.05 60.00)(0.49 68.50)
Expected quality:  55.34          Probability q or greater:  0.66

Cost distribution: (1.00 2.00)
Expected cost:  2.00              Probability c or lower:  1.00

Finish time distribution: (0.09 9.00)(0.35 11.00)(0.03 11.66)(0.47 12.00)(0.05 15.00)
Expected finish time:  11.50      Probability d or lower:  0.45
```

Schedule B - For a client interested in a free solution.

```
        ------------------------------------------------------------------------------------
        | Query-and-Extract-PC-Connection | Query-and-Extract-PC-Mall | Query-and-Process-ZDnet |
        ------------------------------------------------------------------------------------

Quality distribution:
(0.02 0.00)(0.02 8.50)(0.01 10.00)(0.07 18.50)(0.16 28.50)(0.07 30.00)(0.65 38.50)
Expected quality:  33.29          Probability q or greater:  0.65

Cost distribution: (1.00 0.00)
Expected cost:  0.00              Probability c or lower:  1.00

Finish time distribution: (0.11 6.00)(0.38 8.00)(0.04 8.45)(0.31 9.00)(0.17 10.00)
Expected finish time:  8.4        Probability d or lower:  0.52
```

Schedule C - Client requests an even trade-off between quality, cost, and duration.

```
        ------------------------------------------------------------------
        | Query-and-Extract-PC-Connection | Query-and-Process-ZDnet |
        ------------------------------------------------------------------

Quality distribution: (0.20 0.00)(0.08 10.00)(0.72 30.00)
Expected quality:  22.40          Probability q or greater:  0.72

Cost distribution: (1.00 0.00)
Expected cost:  0.00              Probability c or lower:  1.00

Finish time distribution: (0.15 4.00)(0.30 5.00)(0.35 6.00)(0.20 7.00)
Expected finish time:  5.60       Probability d or lower:  0.45
```

Schedule D - Client wishes to maximize quality while meeting a hard deadline of 7 minutes.

```
        ------------------------------------------------------------------
        | Query-and-Extract-PC-Mall | Query-and-Process-Consumers-Reports |
        ------------------------------------------------------------------

Quality distribution:   (0.10 0.00)(0.29 8.50)(0.61 38.50)
Expected quality:  25.88          Probability q or greater:  0.61

Cost distribution: (1.00 2.00)
Expected cost:  2.00              Probability c or lower:  1.00

Finish time distribution: (0.09 5.00)(0.09 5.50)(0.72 6.00)(0.01 6.05)(0.09 8.00)
Expected finish time:  6.05        Probability d or lower:  0.90
```

Figure 4: Custom Schedules for the IG Task Structure

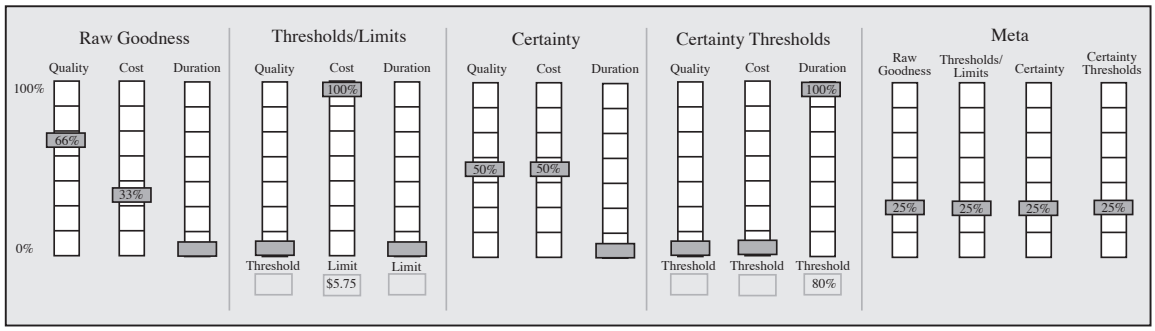

Figure 5: Scheduler Design Criteria Metaphor

As mentioned earlier, the BIG agent uses the RESUN interpretation planner to track and resolve uncertainties about the gathered information and generated hypotheses during information gathering. However, RESUN does not plan internally using the TAEMS modeling framework, in fact, RESUN's opportunistic/blackboard style of problem solving is at odds with the planned-view taken by TAEMS and needed by the scheduler to address resource limitations. The interface between RESUN and TAEMS/DTC is accomplished by a different reasoning component called the task assessor. The task assessor is responsible for formulating the initial information gathering plan and for revising the plan as new information is learned. It manages the high-level process-centered view of the information gathering activities and describes this process via TAEMS for the DTC scheduler. The scheduler, in turn, analyzes the task structure and decides on a course of action for the agent. There is a constant tension between RESUN's opportunistic control and the end-to-end view required to meet deadline and cost constraints. While the details of how this balance is maintained are beyond the scope of this paper [23], the important point is that the task assessor serves as the TAEMS-emitting intermediary between RESUN and DTC.

2.1 Precision versus Coverage in BIG

Thus far we have illustrated Design-to-Criteria's abilities and TAEMS using a small example. Figure 6 shows a complete TAEMS task structure as emitted by the task assessor. Note that there are many more decision points or opportunities for choice in this task structure than in the sample structure of Figure 3. This is how BIG can adjust problem solving not only to meet time and cost limitations, but also to balance between solution precision and solution 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. BIG achieves this trade-off behavior in

two ways: by planning and scheduling for it a priori, and by responding opportunistically to the problem solving context within the constraints of the schedule.

Scheduling for the precision / coverage trade-off is accomplished by relating the precision and coverage specification to quality for the Design-to-Criteria scheduler and giving the scheduler a set of options, from which to choose a course of action. In Figure 6, Get-Extra-Information has two subtasks, Get-More-Objects and Detail-Product-Information denoting the two different ends of the spectrum. Get-More-Objects represents the coverage end and Detail-Product-Information represents the precision end. The sum() quality accumulation function under the parent task, Get-Extra, models the idea 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, i.e., the actions pertaining to precision receive higher quality when increased weight is given to precision. This approach enables the scheduler to reason about these extra activities and their value, and relate them to the other problem solving options from a unified perspective. Thus, the overall value of pre-allocating "extra" time to coverage or precision is also considered in light of the other candidate activities.

BIG can also work opportunistically to improve coverage or precision. While the details are beyond the scope of this paper [23], the general idea is that scheduling often pertains to activities that represent one or more actions in the RESUN planner. This means that within each TAEMS method, RESUN has some flexibility to respond to the current problem solving context or to emphasize coverage or precision. 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.

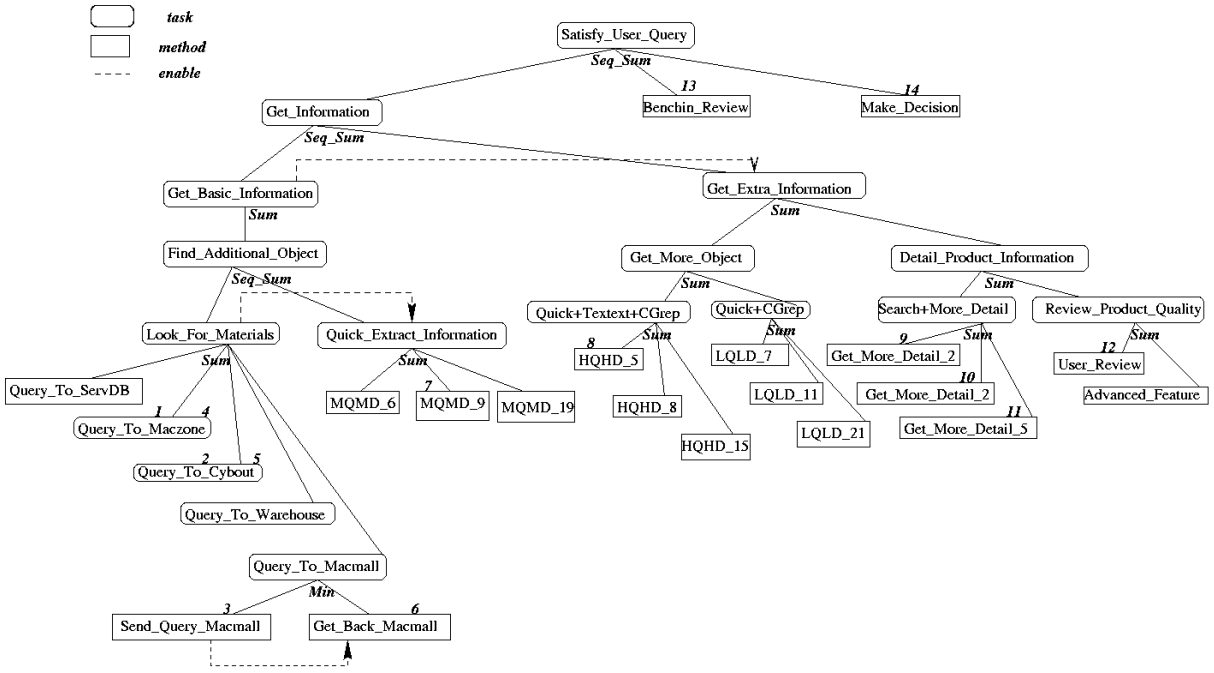

Figure 6: Actual BIG IG Task Structure

2.2 Scheduling for Hard Deadlines

Design-to-Criteria employs a complex set of approximations, heuristics, and satisficing methodologies to cope with the high-order combinatorics of the TAEMS scheduling problem. 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

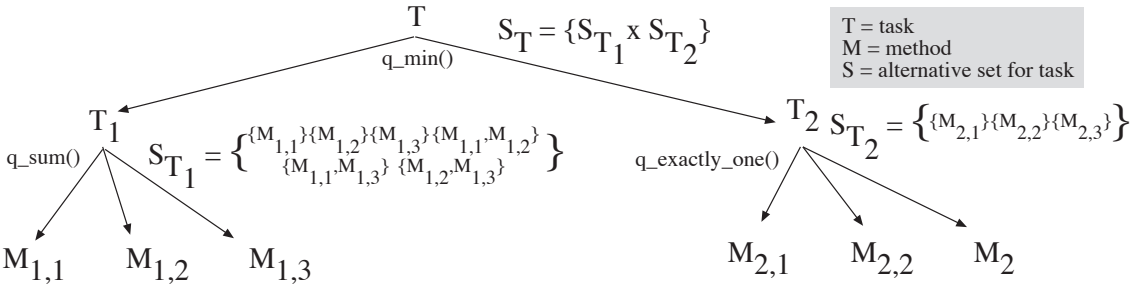

Figure 7: Alternative Sets Lead to Cumbersome Combinatorics

scheduling algorithm must prune schedule approximations, called alternatives, and then 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 7 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 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 set of alternatives for the high-level task is constructed, a subset of alternatives are selected for scheduling.

For situations in which there are 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 7: 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 pre-process 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 set of root level alternatives that is less-focused 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 while still producing good schedules.

3. Cost Based Information Gathering Experiments

To study how BIG address cost limitations, a series of experiments were run. The experimental environment was set up as follows. To simulate widespread fee-for-access models, eight WWW sites that are often used by BIG are arbitrarily assigned different qualities and costs:

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

All other sites are assumed to be free and their quality is determined by the relative number of external references to them (external sites that link to them). The assumption is that the more frequently a site is referred to, the higher its quality is likely to be. Methods making queries to these sites are also associated with different quality and cost characterizations, based on the quality and cost of associated with the site they access.

3.1 Different Schedules Based on Cost

The following experiments show examples of how the BIG system constructs different schedules, which achieve different decisions and have varying characteristics, when given different cost constraints. The sample query to BIG is to find word-processing software for the Windows platform, given an end-to-end completion time of 10 minutes and a price limit of $200 for the final product. The

DTC scheduler was used with the following objective function or design criteria, which as noted above dictates how it should value tradeoffs between alternatives. The relative importance of the three criteria for the searching process is specified as follows: quality, 85%; cost, 10%; duration 5%. This setting implies quality should be the most important criterion when evaluating schedules. The relative importance of the cost limit and the time deadline is 90% and 10%, respectively — the cost limit is far more important than the duration limit. These specifications tell the DTC scheduler to search for a schedule that achieves the highest quality within the cost limit. However, the time deadline may not be met in every instance because the duration limit is less important.

Given different searching cost criterion, the scheduler generates different schedules that try to stay under the given cost threshold while also attempting to maximize time and quality objectives. When given cost thresholds of $0, $1, $3, $6, $9, different schedules are constructed, as shown in Table 1. In this table, a "Y" in a row indicates that the method in that row is included in the schedule with a specific cost threshold. For example, method Query_To_Netsales is only used in the schedule where the cost threshold is nine dollars.

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

total cost		0	0.8	2.8	6	8.2	
total quality		565	653	776	1277	1502	

Table 1: BIG schedules under varying cost thresholds.

When the search cost of threshold is $0, the BIG system first makes a query to the free site "zdnet" to get basic product information, it then chooses seven text-processing methods to process information (these methods are also free). It then elects to query a free review site, "cybout review," to get review information. Finally, it makes a decision based on the available information. In contrast, when given the search cost threshold is $1, the system makes an extra query to the "pczone" site that charges $0.8 and gains 112 quality units from this method. When the search cost threshold increases to $3, the system spends $2 on the "benchin" review site and earns 189 quality units from this action. Given $6, the system queries three product sites: "zdnet," "pczone," and "pcmall." It also queries the "benchin" site twice for different review information: company reviews and user reviews. When given a searching cost of $9, the system chooses to query all five product sites and three review methods. In all five cases, the method "Review_Finder" is not chosen because its quality/cost ratio is lower relative to other alternatives.

3.2 Experimental Results

We ran experiments with five different given cost thresholds, $0, $1, $3, $6, and $9. For each specific cost, we ran the system 10 times and present the experiment results in the Table 2.

The first column and the second column are the user-specified cost limit (S.C.) and the real searching cost (R.C.). The real searching cost is very close to the user request cost limit without exceeding the limit because the scheduler was directed to put great importance (90%) on not going over the cost threshold. The next four columns denote the number of considered products (C.P.), total number of products found (T.P.), aggregate information coverage (I.C.), and average information coverage per product object (A.C.). These values reflect the number of information sources used to generate the final decision. Given additional cost, BIG will adjust its searching behavior in an attempt to find both more sources of information, and more supporting information for previously discovered products. The number of products considered and the total number of products found (T.P.) did not increase from the experiment with a cost of $1 to the experiment with a cost of $3. In both cases, BIG chose to spend the extra $2 on a high-quality review site to get better review information on previously discovered products, which is consistent with the semantics behind the objective criteria we supplied. The next column is the information quality (I.Q.), which reflects the quality of the information sources that contributed to the decision. The information quality increases according the searching cost because high-quality sites charge a higher fee than lower quality sites. The next column denotes the extraction processing accuracy per object (P.A.),

supplied in part by the information processing tools. This characteristic is not affected by the searching cost, because our text processing tools are cost independent. Decision confidence (D.C.), generated by the decision maker, reflects the likelihood that the selected product is the best choice among the entire set of products considered. This value is based on the quality distributions of each product, and represents the chance that the expected quality is correct. This value is not directly related to search cost thresholds. The scheduling time (S.T) for the first three experiments is very close to the given deadline. The last two experiments exceed the deadline because the scheduler has significantly more money to spend, and therefore can expect to obtain a higher quality decision, so it chooses to spend more time. The execution time (E.T.) increases with cost because the increased amount of money spent makes it more likely that queried sites will contain relevant material. This in turn will increase the overall time it will take to extract relevant information, since there will be more documents to process. The final column shows the more frequently chosen product as the final decision in the set of ten runs. Overall, the results indicate that the more money a client is willing to spend in gathering high quality information, the higher the quality will be of the final product selection decision BIG makes -although clearly this trend cannot continue forever (in the final two runs the same product was chosen). Note also that it is quite possible to find the best product without spending any money at all, in this example we show that it is just more likely that a better product can be found as a result of the higher quality evidence gathered because more money was spent.

	S.C.	R.C.	#C.P.	#T.P.	I.C.	A.C.	I.Q.	P.A.	D.C.	S.T.	E.T.	final decision
average	0	0	1	3	8	1	0.33	1.83	1	667	133	product name: Lotus FastSite 2.0
st.dev.		0	0	0	0	0	0	0	0	0	2.2	price: 99
												occurrence: 10/10
average	1	0.8	8	22	28	1	0.57	1.08	0.84	600	414	product name: Lotus Word Pro 97
st.dev.		0	0	0	0	0	0	0	0	0	35.4	price: 63.98
												occurrence: 6/10
average	3	2.8	7	22	33	1.71	0.64	0.96	0.88	670	470	product name: Lotus Word Pro 97
st.dev.		0	0	0	0	0	0	0	0	0	11.3	price: 63.98
												occurrence: 8/10
average	6	6	15.2	54	67.5	1.5	0.83	1.16	0.83	990	1041	product name: Word Pro 97 CD WIN/W95
st.dev.		0	0.87	0	1.96	0.1	0	0.03	0.06	0	50.5	price: 59.19
												occurrence: 8/10
average	9	8.2	21.6	69.8	80.5	1.44	0.92	1	0.85	974	1134	product name: Word Pro 97 CD WIN/W95
st.dev.		0	1.36	0.6	0.5	0.01	0.02	0.05	0	0	52.1	price: 59.19
												occurrence: 10/10

Table 2: Experimental results when running BIG with different cost thresholds

4. Conclusions

In this article, we have discussed how a sophisticated information gathering agent, BIG, can be organized so that it can take into account complex resource-bounded constraints on its activities. We have detailed how the domain-independent Design-To-Criteria scheduler can be integrated into the architecture so it generates appropriate schedules based on complex client criteria. We have also shown in experimental results that the ability to access costly, but high quality, information

WWW sites can produce a higher quality decision process. More generally, we have indicated that it is feasible to begin to develop complex information gathering agents that can adjust their behavior based on available processing and monetary resources. We feel that as the trend develops to charge for accessing valuable information, i.e. an information marketplace, and agents develop the ability to not only retrieve documents but also process their contents, the capabilities we have explored in BIG will become the norm. We also feel that the basic architecture we have laid out for resource-bounded reasoning will also be appropriate when the prices for accessing information may involve more sophisticated negotiation. In this case, the scheduler can be used to assess what the expected effects of certain information costs on the overall decision process of an agent. This analysis is a key component in the agent deciding what would be a reasonable price to pay for specific information and comparing alternative bids from other sites which may have different quality and coverage attributes.

5. Acknowledgments

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

6. References

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

[2] 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.

[3] Norman Carver and Victor Lesser. A planner for the control of problem solving systems. *IEEE Transactions on Systems, Man and Cybernetics, Special Issue on Planning, Scheduling and Control*, (6): 1519-1536, 1993

[4] 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 Multi-Agent Systems*, June 1995.

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

[6] Keith S. Decker and Victor 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 in UMass CS-TR 94-14.

[7] K. Decker, V. 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.

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

[9] Stanley M. Sutton Jr. and Leon J. Osterweil. The Design of a Next Generation Process Language. Umass CS TR-96-35.

[10] E. Durfee, T. Mullen, S. Park, J. Vidal, P. Weinstein, Strategic Reasoning and Adaptation in an Information Economy. In Intelligent Information Agents (M. Klusch, ed.), Chapter 8, pp. 176.203. Springer-Verlag, 1999.

[11] 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.

[12] Joshua Grass and Shlomo Zilberstein. Value-Driven Information Gathering. In *Proceedings of the AAAI Workshop on Building Resource-Bounded Reasoning Systems*, Providence, RI, 1997.

[13] B. Horling, V. Lesser, R. Vincent, A. Bazzan, P. Xuan, (1999). Diagnosis as an Integral Part of Multi-Agent Adaptability. University of Massachusetts/Amherst Computer Science Technical Report 1999–03, January.

[14] 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 1997

[15] Adele Howe and Daniel Dreilinger. A Meta-Search Engine that Learns Which Engines to Query. *AI Magazine*, 19(2), 1997.

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

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

[18] D. Jensen, T. Oates, and P. Cohen, Correlating and Predicting Asynchronous Events." AAAI 1998 Workshop on Predicting the Future: AI Approaches to Time Series

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

[20] V. Lesser, M. Atighetchi, B. Benyo, B. Horling, A. Raja, R. Vincent, T. Wagner, P. Xuan, and S. Zhang, (1999). A Multi-Agent System for Intelligent Environment Control. *Proceedings of the Third International Conference on Autonomous Agents*, Seattle, pp. 291-298.

[21] 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.

[22] 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 National 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.

[23] V. Lesser et al, "BIG: A Resource-Bounded Information Gathering and Decision Support Agent," Under review, a version is also available as UMASS CS TR 1998-52

[24] T. Oates, M.V. Nagendra Prasad, and V. Lesser (1997). Cooperative Information Gathering: A Distributed Problem-Solving Approach. In *IEEE Proceedings on Software Engineering*, Special Issue on Agent-based Systems, Volume 144, No. 1, pp 72-88.

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

[26] Charles Rich and Candace L.Sidner. Collagen: When agents collaborate with people. *In Proceedings of the First International Conference on Autonomous Agents* (Agents 97), pages 284-291, 1997.

[27] 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.

[28] Ion Muslea, Steve Minton, and Craig Knoblock. A hierarchical approach to wrapper induction. IN Proceedings of the Third Annual Conference on Autonomous Agents, pages 190-197, Seattle, WA, May 1999. ACM Press.

[29] R. Vincent, B. Horling, T. Wagner, and V. Lesser, (1998.) Survivability Simulator for Multi-Agent Coordination. In *Proceedings of International Conference on Web-Based Modeling and Simulation*, SCS (eds.).

[30] 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.

[31] 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.

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

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

[34] David Jensen, Michael Atighetchi, Regis Vincent, Victor Lesser. Learning Quantitative Knowledge for Multiagent Coordination. In *Proceedings of the*

Sixteenth National Conference on Artificial Intelligence, pages 24-31, July 1999.