Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

UMI®

VALUE-DRIVEN INFORMATION GATHERING

A Dissertation Presented

by

JOSHUA W. GRASS

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

February 2002

Computer Science

UMI Number: 3039360

UMI Microform 3039360

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

VALUE-DRIVEN INFORMATION GATHERING

A Dissertation Presented

by

JOSHUA W. GRASS

Approved as to style and content by:

Shlomo Zilberstein, Chair

W. Bruce Croft, Member

Victor Lesser, Member

Richard Giglio, Member

W. Bruce Croft, Department Chair
Department of Computer Science

DEDICATION

To Holly, Henry and Patricia

ACKNOWLEDGMENTS

I would like to thank my advisor Shlomo Zilberstein for his advice, guidance and patience over the past several years. I would also like to thank my committee members: Victor Lesser, Bruce Croft and Richard Giglio for their feedback and questions throughout the project.

I also want to thank three other teachers that I had over the years: Ronald Tenison, Eddie Clark and Kris Hammond. Their support and guidance was invaluable during high-school and college.

Finally, a special thank you to all my friends, co-workers and family for their support and encouragement over the years.

ABSTRACT

VALUE-DRIVEN INFORMATION GATHERING

FEBRUARY 2002

JOSHUA GRASS, B.S., UNIVERSITY OF CHICAGO
M.S., UNIVERSITY OF MASSACHUSETTS, AMHERST
Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Shlomo Zilberstein

This dissertation addresses the problem of autonomous information gathering from a large distributed network of information sources. Information gathering is viewed as a component of a decision support system, which uses a set of rules and a set of information sources to recommend an action. This recommendation includes a prediction of the utility for selecting this action and the level of confidence that the system has in the decision. A decision support system has two primary tasks when making a well-informed, well-reasoned decision. The first task is to gather information about the state of the world that is relevant to making the decision; and the second task is to use this information and a set of rules to evaluate a set of potential actions and make a recommendation. A large number of information gathering systems have been developed in recent years that use the Internet as their primary source of information. However, the overwhelming amount of information available on the Internet has created a new problem for information gathering systems: it is no longer feasible to query and process all of the available relevant information. Next-generation information gathering systems must account for the resources required to query and process the information sources used by the system.

To address this problem, this dissertation develops a decision-theoretic framework for information gathering that is sensitive to several characteristics of information sources.

These characteristics include the value of acquiring a piece of information with respect to the specific user's decision model, the strength of the evidence returned by the information source, the immediate cost of querying the information source, and the expectation of when and if the query will return information. The comprehensive value of a query, which is an extension of the decision-theoretic notion of the value of perfect information (VOI), is calculated using these characteristics. Much like the VOI, the value of a query is based on the notion of determining the expected increase in the overall expected utility of the decision as a result of issuing the query. However, unlike the VOI, the value of a query reflects the fact that information gathering is not instantaneous and may have associated costs.

There are three main contributions made by this dissertation. The first contribution is the development of a formal framework for query planning with limited information gathering resources that is driven by the user's decision model (an influence diagram). The second contribution is implementing this framework as an expandable system for creating autonomous information gathering agents. The third contribution is demonstrating how value-driven query planning yields improved information gathering strategies that return high-quality decisions while using substantially fewer resources.

As the number of information sources available to autonomous information gathering systems grows, the role of reasoning about both the cost and benefits of querying any potential information source becomes increasingly important.

TABLE OF CONTENTS

Page

ACKNOWLEDGMENTS v

ABSTRACT vi

LIST OF TABLES xii

LIST OF FIGURES xiii

CHAPTER

1. INTRODUCTION 1

1.1 Information Gathering 1
1.2 The environment of information sources 3

1.2.1 Redundant information sources 5
1.2.2 Querying cost 6
1.2.3 Responsiveness 6
1.2.4 Uncertainty 6

1.3 The Internet as an information source environment 7

1.3.1 Redundant information sources 7
1.3.2 Querying cost 8
1.3.3 Responsiveness 8
1.3.4 Uncertainty 9
1.3.5 Extraction 10

1.4 Value-driven Information Gathering Applications 11

1.4.1 Product selection 12
1.4.2 Medical diagnostic systems 13
1.4.3 Quality testing 13
1.4.4 Personnel Management 14

1.5 Value-driven information gathering 15

1.5.1 Thesis 16

1.6 Outline 21

2. BACKGROUND 23

2.1 Decision Models for Reasoning under Uncertainty 23

2.1.1 Incorporating evidence into belief networks 31

2.2 Resource-Bounded Reasoning 35
2.3 Information Extraction & Integration 41
2.4 Integrated systems 45

2.4.1 High-level information agents 45
2.4.2 Information Extraction from Heterogeneous Sources 47
2.4.3 Design-to-criteria information gathering 48
2.4.4 Cooperative information gathering and processing 51
2.4.5 The Ariadne project 52

2.5 Recursive information gathering plans 53
2.6 Discussion 55

3. PROBLEM STATEMENT 57

3.1 Decision Model 59

3.1.1 Prior probability distributions 63
3.1.2 Instances 64
3.1.3 Evidence Nodes 68
3.1.4 Utility score 73

3.2 Cost function 76
3.3 Information sources 79
3.4 Gathering session 82

3.4.1 Querying constraints 86
3.4.2 The value-driven information gathering problem 86

4. VALUE-DRIVEN INFORMATION GATHERING 88

4.1 Value-Driven Information Gathering Algorithm 89
4.2 Determining the value cf a query 91

4.2.1 Optimizations to calculating the value of a query 97

4.3 Determining the value of finding a new instance 98

4.4 Limited number of outstanding queries 99

4.4.1 Optimizations for finite outstanding queries 101

5. COMPONENTS OF A VALUE-DRIVEN SYSTEM 103

5.1 Decision model 103
5.2 Information sources database 104
5.3 Communication layer 105
5.4 Value-driven information gathering planner 105
5.5 Implementation 107

6. EXPERIMENTAL RESULTS 108

6.1 Testing methodology 108

6.1.1 Comparison systems 108
6.1.2 Evaluation metrics 111
6.1.3 The Simulated Internet Environment 111

6.2 Sequential querying 112

6.2.1 Domain: Car purchasing 114
6.2.2 Evaluation 114

6.3 Parallel querying 115

6.3.1 Domain: Job evaluation 116
6.3.2 Evaluation of Job Move decision 118
6.3.3 Domain: Software purchasing decision 119
6.3.4 Evaluation of Software purchasing decision 120

6.4 Implementation on the Internet 121

6.4.1 Domain: Digital camera selection 122
6.4.2 Evaluation 124

6.5 Adding evidence and instances 126

6.5.1 Domain: Removable media purchase 127
6.5.2 Evaluation 128

6.6 Dynamically altering the decision model 130

6.6.1 Domain: Restaurant selection 135

6.6.2 Evaluation 135

6.7 Trace 136

7. CONCLUSIONS 141

7.1 Summary of contributions 141
7.2 Limitations and extensions 143

7.2.1 Scalability 143
7.2.2 Combining with other research 144
7.2.3 Internal resource allocation 144
7.2.4 Myopic planning 145

7.3 Future directions 145

BIBLIOGRAPHY 147

LIST OF TABLES

Table **Page**

Table 1.1 – Variance in the information gathering session based on the result of one query returning different results ... 9

Table 2.1 – Conditional probability table for the node *Afternoon rain* (see **Figure 2.1**) ... 25

Table 2.2 – Probability table for *Morning weather* (see **Figure 2.1**) ... 25

Table 2.3 – The utility table for the afternoon rain influence diagram (see **Figure 2.3**) ... 28

Table 2.4 - The probability distribution for each combination of the cut set ... 39

Table 6.1 - Results of the car purchasing experiments ... 114

Table 6.2 - A comparison of different information gathering strategies for the job evaluation experiments ... 119

Table 6.3 - A comparison of different information gathering strategies for the software purchasing experiments ... 120

Table 6.4 - Comparison of VDIG to other retrieval methods for the digital camera experiments ... 125

Table 6.5 - The products, features, and web sites used in the removable-media evaluation process ... 128

Table 6.6 - Results of value-driven information gathering ... 129

Table 6.7 - Results of restaurant decision system ... 135

LIST OF FIGURES

Figure Page

Figure 1.1 – Value-driven information gathering approach versus an information gathering system based only of the value of information.......... 20

Figure 2.1 - A simple belief network with three variable nodes.............................. 24

Figure 2.2 – Data required to represent the input and output states of a probabilistic network using the belief network approach and a black box approach.. 26

Figure 2.3 – An influence diagram constructed from the simple three node belief network (see **Figure 2.1**) 27

Figure 2.4 – Adding a new node to a Belief Network... 32

Figure 2.5 – A Noisy-Or node... 33

Figure 2.6 – Algorithms with a satisficing set of solutions...................................... 36

Figure 2.7 - The cut set for any node in an acyclic belief network......................... 38

Figure 2.8 - An example TAEMS task structure for Gathering Auto purchase information... 49

Figure 3.1 - An overview of the value-driven information gathering process......... 57

Figure 3.2 – Digital camera decision model 60

Figure 3.3 - Digital camera belief network for one instance.................................... 61

Figure 3.4 - Restaurant decision model.. 62

Figure 3.5 - Restaurant instances and evidence 65

Figure 3.6 - Digital camera belief network 67

Figure 3.7 - Evidence collected for the image size 68

Figure 3.8 - Generating a probability table from an accuracy and bias.................... 69

Figure 3.9 - Conditional probability tables for different types of evidence nodes 71

Figure 3.10 - Restaurant utility function 74

Figure 3.11 - Sample cost of time function with a linear monetary cost and an exponential time cost 76

Figure 3.12 – Sample cost of time functions with a "grace" quantity of time and money, which the gathering session may use with no resource cost...... 77

Figure 3.13 – Sample cost function with a deadline time 78

Figure 3.14 – Restaurant resource cost function 79

Figure 3.15 - Information source response-function used in value-driven information gathering 80

Figure 3.16 - Restaurant information sources database 81

Figure 3.17 - Expected utility curve over time 82

Figure 3.18 – The expected utility function over the course of the gathering session (no additional queries made beyond the initial set at time zero)...... 83

Figure 3.19 – Expected utility function during a query returning and an new query being made 85

Figure 4.1 - The VDIG algorithm 89

Figure 4.2 - Expected utility curve over time 90

Figure 4.3 – The state of the potential query set (P), the query pool (Q) and the returned evidence (R) during a gathering session 92

Figure 4.4 - The value of information for multiple information sources may not be additive 95

Figure 5.1 - The components of a value-driven information gathering system 103

Figure 5.2 - The state of the gathering session after an information source and an instance source have returned 105

Figure 6.1 - Job decision influence diagram 117

Figure 6.2 - The value-driven information gathering system running for the job evaluation experiment 118

Figure 6.3 - A response-function for an information source in the job evaluation experiment 119

Figure 6.4 - Software purchasing influence diagram 120

Figure 6.5 - Expected utility including the extraction time for the digital camera experiment 121

Figure 6.6 - Digital camera decision model 122

Figure 6.7 – The VDIG working on a camera decision 123

Figure 6.8 - The response-function for an information source in the camera experiments 124

Figure 6.9 - The expected utility curve for a run of the removable media experimental system 127

Figure 6.10 - Decision quality over time for the removable media experiments... 130

Figure 6.11 - Decision model for each instance of a restaurant 132

Figure 6.12 – At the start of the information gathering session 136

Figure 6.13 – After the initial set of queries have been made 137

Figure 6.14 – State of the gathering session after an information source and an instance source have returned 138

Figure 6.15 – The end of the information gathering session 140

CHAPTER 1

INTRODUCTION

1.1 Information Gathering

Individuals and groups make many decisions every day. These decisions may be as simple as whether to bring an umbrella to work or as complex as deciding whether to bring a new product to market. A decision support system uses three sets of data when making a decision: a set of rules, which are used to evaluate, or score, a set of potential actions; and the set of information items used by those rules. All of the rules and actions are known by the decision support system at the beginning of the evaluation, but the specific items in the set of information must be gathered or collected by the system over the course of the evaluation. These three classes of data are referred to as *rules*, *actions*, and *evidence*. A decision support system collects evidence, which instantiates rules and influences the expected utility of selecting an action.

A piece of evidence can have varying influence on our decision; some pieces may have a small amount of influence, while others may make nearly all other information irrelevant. In our umbrella example, seeing a cloudy sky is a piece of evidence that strongly encourages us to bring an umbrella, while a report on the radio might have a smaller impact on the decision. In order to make good decisions a person needs both a good set of rules and the information to use those rules. *Information gathering* is the process of collecting information in order to make a decision. As long as our set of rules are consistent and accurately model the problem, the quality of our decision will increase as we gather more information.

One example is a common event in the business world: one company (Alpha) is given the opportunity to purchase another company (Beta). The decision to purchase Beta is based on several factors: the price, the revenue of the company, its profits, the staff, and its inventory. A decision to purchase Beta could be made at any time, but the confidence that Alpha has in the decision will increase as it gathers more information. Of course, there is also a point when more information begins to have less and less of an effect on the decision. If for example, Alpha learns early on that Beta will cost much more than it can afford, gathering more information will not alter the final decision to not purchase the company.

It is also important to consider the resources that are required for Alpha to learn about Beta. Gathering this information has two costs: a monetary amount to hire or interview the correct people, and a cost in time for the information to be gathered and incorporated into the decision. Much of the information needed in order to make a good acquisition decision for Alpha is not easy to find, experts must be hired to analyze the books of Beta and judge its health, people in Beta need to be interviewed and the property of the company needs to be inspected. Gathering each of these pieces of information is an action that has a cost to initiate and an expectation of when the results will be returned.

Finally, Alpha must consider that the time used to make the decision is not free. Other companies may have also been offered the opportunity to purchase Beta; the resources used to make the acquisition decision may be better spent working on other problems; and the business environment itself might change if too much time is spent evaluating Beta. Time, much like money, is a limited resource that cannot be wasted.

The company purchasing example illustrates how important information gathering is in the process of making an effective decision. No matter how good the rules are for making a decision, a good information gathering strategy is also important for the overall effectiveness of the decision making system. This dissertation will focus on how to gather information effectively in order to make high-quality decisions using limited resources.

The goal of this work is to define the process of information gathering done by decision support systems and describe techniques for effectively planning the information gathering process. The process of gathering information is as important as to the overall quality of the decision made by a decision support system as the set of rules used to evaluate the information. The dissertation will demonstrate that if a decision support system effectively takes the resource cost of gathering information into account, it can return substantially better decisions than a system that does not consider these factors, even if these two systems have the same set of rules for selecting an action. Decision support systems that effectively gather information will also be able to use the resources they save for other tasks. This dissertation develops a class of decision support systems, which incorporate the resource cost of information gathering into the decision making process. The process of including both the value of information and the resource cost of gathering that information is called *value-driven information gathering (VDIG).*

1.2 The environment of information sources

Value-driven information gathering operates in an environment of *information sources*. An information source is an abstraction for any item, action, or event that can return information (evidence) used in the decision. In order to gather information, the decision support system must *query* an information source. Querying an information source has a comprehensive cost that is based on the amount of time that the information source will take to return and the initial cost of making the query. The initial cost may include a monetary fee for accessing the information source, a computational cost for processing the query, or both. The amount of time required for a query to return is called the *response* or *response-function* of the query and is defined by a function that returns the probability that a query will have returned the evidence by time t.

An information source is defined as external for two reasons: First, the agent has no control over how long it will take the information source to return the results of a query; and second, the number of active queries has no effect on the behavior of the value-driven system. Queries are not analogous to a separate computational process on the same processor because they do not affect the computational resources available to the decision support process. In our company purchasing example, we can decide whether to send an inspector to evaluate a factory or not, but we have no control over how long the inspection will take.

In some cases, we may wish to relax the restriction that the decision support system cannot control the response of an information source once it is queried. The most obvious case where this might occur is when a decision support system could query an information source with a higher initial cost in exchange for a lower expected time for the query to return a result. This can be represented by a set of different information sources with different cost and response-functions. When one of the choices is selected, the other queries to the same information source are removed from the list of potential queries. As long as the system is restricted to a finite number of initial-cost / response choices, this approach will work. To expand on our factory inspector example, the company may also have the opportunity to send a team of inspectors that would cost more monetarily, but would also return the evaluation of the factory in less time.

In many situations, there is an over-abundance of potential information sources that can be used to make a decision. The decision maker does not have the resources, or the need, to query all the available information sources. Instead, the agent has the problem of determining which information sources will influence the decision the most and use the fewest resources.

Before the development of the World-Wide-Web, the development of autonomous information gathering systems has been examined in detail [72]. Most system have assumed that information sources are homogeneous (i.e. all taking around the same amount of time to return) and distinct (few alternatives that would return similar information). In the environment used by value-driven information gathering systems, it is assumed that the decision making system will have access to a large number of information sources that vary greatly in the information they return and their responsiveness. The information sources that the system can query also have a large amount of redundancy in the information they return to the system. For example, information source **A** may return information item **1**, **2**, and **3**, and information source **B** may return information item **3**, **4**, and **5**. At the beginning of developing value-driven information gathering, these redundant sources were assumed the same, latter in the research we relaxed this restriction and modeled information items returned by information sources as pieces of evidence that can conflict or support each other.

Deciding which information to gather in making a decision is a process that we as humans do constantly. When a person looks both ways before making a decision to cross a street, each of these observations is an information gathering action that takes time, but the pay-off is that we make a better street crossing decision. Hidden in this example is a large amount of control and planning. Why don't we take a second look to the right? Why do we sometimes look again halfway across the street? Even this simple task tells much about our view of the world, namely, how quickly the world changes, and how important it is not to make a mistake.

Systems, which spend time considering the resource cost of making a query as well as the information that will be returned by the query, are more effective at planning with limited resources. In very few cases is information free (when we consider time as a cost), and the results from querying an information source are almost never returned instantaneously. In complex environments, we must ask ourselves several questions about the information sources that we plan on querying:

- How useful is the information in making the decision?
- How quickly can this information be retrieved?
- How accurate is the information?
- How quickly does this information change?

As the number of potential information sources in the environment increases, information gathering systems have a better chance of finding information sources that are a better "fit" with the information needed and the resources available. For systems that can evaluate the impact of the responsiveness, accuracy and costs of a query, having a larger selection of information sources generally means the quality of the decision will increase. The more redundancy and choice there is in the set of information sources, the better it is for the system.

The algorithms described in this dissertation were designed to operate in environments in which there is a large amount of redundancy and variability in the information sources, which can be queried by the decision making system. Listed below is a description of the environmental characteristics that have the greatest impact on value-driven information gathering:

1.2.1 Redundant information sources

The more information sources are available, the better a value-driven information gathering system will perform. Overlapping fragments of information from multiple sources allow the value-driven system to find an information source that most closely matches the information needed using the resources the system has to spend.

1.2.2 Querying cost

There should be some cost or limit to the number of sources one may query. Otherwise querying every information source available immediately is an optimal strategy. Conversely, querying an information source should not be prohibitively expensive, or the benefits of querying information sources in parallel decreases.

1.2.3 Responsiveness

Information sources should vary in their responsiveness. The variance allows the system to pick information sources that closely match the resource restrictions imposed on the system. In order to effectively schedule queries, the system must have reliable estimates of the responsiveness of the information sources. These estimates do not need to have a high degree of accuracy, but as the accuracy increases, so does the effectiveness of the value-driven information gathering approach. For the experimental cases described in Chapter 6, the simple approach of randomly sampling sites on the web was adequate to generate response-functions that the value-driven information gathering systems could use to differentiate the information sources. Only in cases in which the true response-function of an information source was drastically in error, *compared to most of the other information sources*, would this have an effect on the behavior of the system. Errors that have global effects on the response-functions for all of the information sources, such as variance in the speed of the Internet connection to the querying machine, have little effect on the behavior of the system.

1.2.4 Uncertainty

Uncertainty in the responsiveness of an information source reduces the benefits of long-term planning. Often times, the information returned by a source can drastically change the value of future queries. Uncertainties in the response time reduce the benefits of generating a gathering plan and make a reactive system a better choice.

Extraction

Value-driven information gathering requires that the data returned by an information source can be extracted and incorporated into the decision model.

If these characteristics of the environment are relaxed, a value-driven approach will continue to work, but so will less computationally expensive information gathering strategies.

1.3 The Internet as an information source environment

This dissertation focuses on the Internet as an environment for gathering information for use by decision-making systems. Therefore, it is not surprising that the environment in which value-driven information gathering works best is based in large part on the characteristics of the Internet.

Besides providing a challenging environment for gathering information, the Internet has many other features that make it a good test-bed for creating decision making agents. The Internet is cheap and easy to access, there is a large amount of content, the space is so large that navigation is not easy, the environment is safe, there is a large amount of research and tools being developed, and it is relatively new domain in need of powerful tools. Because all of the empirical results for this dissertation have been generated using the Internet or simulations of the Internet, it is important to illustrate how the internet meets the assumptions that I have made in designing the value-driven algorithm.

1.3.1 Redundant information sources

The Internet, especially in the product evaluation domain, is full of redundant information sources. For example, most products have at least four sources of information: The manufacturer (company web pages), the distributor (on-line catalogs), professional evaluation sources (web magazines), and amateur evaluation sources (individual home pages or news-groups). Each of these types of information sources has their own advantages and specific types of information they focus on. Also in most cases (with the exception of the manufacturer) there are multiple organizations doing the same thing.

1.3.2 Querying cost

The world-wide web at this time has sites that are free to access (The Boston Globe, product catalogs, CNN) as well as sites, which charge a monetary fee (The Wall Street Journal, The New York Times archives, real-time stock information). Besides the monetary fee for accessing an information source, the computational resource cost of querying a site and extracting the information must also be considered. Creating and monitoring a network socket to a web server uses processing resources, and many computers have a limit on the number of simultaneous connections they can effectively make. Fortunately, the price of querying a site is not so high that the optimal solution is uninteresting. Making a query that ultimately does not contribute much to the decision that is made is not catastrophic and the monetary cost of most pay sites is a tiny fraction of the cost of the product being evaluated. This means that decision making system can query sources that have a high level of variability.

1.3.3 Responsiveness

Servers on the web have a probabilistic chance of returning information at any time after being queried. Although this information can vary depending on regional and local server load, measures of this load can be found on numerous web sites. In addition, it is often the case that several information sources are located on the same server, so once the responsiveness is found for a server, the responsiveness can be generalized across many information sources.

Restaurant querying session 1		
Time	**Action**	**Value**
1 sec	Query	Restaurant 1 price
1.5s	Query	Restaurant 1 cuisine
2.3s	Query	Restaurant 2 price
3.5s	Result	Restaurant 1 cuisine
Restaurant 1 cuisine = Chinese		
4.5s	Query	Restaurant 1 location
5.2s	Result	Restaurant 1 price = low
5.7s	Query	Restaurant 3 cuisine
8.7s	Result	Restaurant 2 price = med
8.9s	Query	Restaurant 3 price
9.9s	Result	Restaurant 3 price = med
10.2s	Result	Restaurant 1 location = med
10.5s	Choose	Restaurant 1

Restaurant querying session 2		
Time	**Action**	**Value**
1 sec	Query	Restaurant 1 price
1.5s	Query	Restaurant 1 cuisine
2.3s	Query	Restaurant 2 price
3.5s	Result	Restaurant 1 cuisine
Restaurant 1 cuisine = French		
4.5s	Query	Restaurant 3 price
5.2s	Result	Restaurant 1 price = low
8.3s	Query	Restaurant 2 location
8.7s	Result	Restaurant 2 price = med
13.2s	Query	Restaurant 3 cuisine
14.6s	Result	Restaurant 3 price = med
16.3	Result	Restaurant 3 cuisine = Chinese
17.7s	Choose	Restaurant 3

Table 1.1 – Variance in the information gathering session based on the result of one query returning different results

1.3.4 Uncertainty

Analysis of servers and sites on the Internet has shown that there is a large amount of variability in the response times for information sources. This variability can change from moment to moment and can have wide variation depending on time of day and other factors. In addition, in many of the product selection problems we studied, there was a broad selection of information sources available. Thus, attempting to generate a long-term information gathering plan was not useful. For example, the list of queries and times at which those queries were made could vary greatly depending on the evidence returned by an information source queried early in the gathering process. **Table 1.1** demonstrates how an information gathering session can vary based on the results returned by an information source early in the session. Both of these sessions are the same until second 3.5 when *Restaurant 1 cuisine* returns different results. Receiving these differing results may change both the final decision made by the system and the future queries made by the system. After the different results, the two querying sessions diverge dramatically. In the example below, because the user has a high preference toward Chinese cuisine, one restaurant 1 responded to a query with the information that is served that type of food, the system quickly focused on choosing that restaurant. Future queries made by the system can also change based on *when* a query returns as well as the information it returns. Two querying sessions in which the same query returns at different times might also diverge. The large number of possible system states based on the information returned, as well as when the information is returned, for all potential information sources, makes creating a general querying plan at the start of the session infeasible.

1.3.5 Extraction

In many ways, extraction is the hardest challenge in using the Internet as a test-bed. Although many information sources have data in an easy to use format (e.g. easily readable in a table), there are also information sources that use natural language to describe the features of a product. In value-driven information gathering the problem of extraction is dealt with in two ways: First, the large amount of redundancy in information sources available on the Internet allows the system to focus on sources that have easily extractable information. Second, several extraction methods described in the current automated extraction research literature has been implemented to increase the number of sites that a value-driven information gathering system can use. Although this dissertation is not on information extraction, it acts as a good proof of concept for some previous information extraction research [4][5][25][18][50].

All of these researchers are working on the creation of automated wrapper systems that convert web pages from a semi-structured form to a form that can be accessed much like a traditional database. The value-driven information gathering system has placed this wrapper layer between the Internet and itself, to solve many of the problems of information extraction. Of course, these tools, like our own, are still in the early stages, but considering the number of people working on automated information extraction I am confident that usable automatic wrapper system will be available in the near future for value-driven information gathering systems to use. In addition, semantic tagging of web pages, using such languages as XML, will simplify the information extraction process in the future [70].

The usefulness of having clear information representations, which can be accessed and processed by computational systems, has led to the development of the semantic web [7][10]. The semantic web is a layer of information that resides below the standard hypertext markup language, which is used by browsers to display information for a user. The semantic web is constructed using XML (eXtensible Markup Language), RDF (Resource Description Framework), and set of ontologies. These technologies are used to construct simple atomic statements, which are embedded in the document, and can be used by an autonomous system to reason and extract data from the document. The ontologies associated with a set of documents contain taxonomy information, definitions of the relationships and equivalence between objects, and rules of inference, which allow the system to generate new data using the existing data. While the set of data in a set of documents created by the same author may be internally consistent, combining and manipulating information from documents created by multiple authors and groups require a set of rules, which can convert terms and explain their relationship, both in the document and across documents. The development of concepts like the semantic web and the technologies to implement it are crucial for value-driven information gathering, and any other complex autonomous system that uses the Internet, to operate effectively and automatically across a large number of information sources. The extreme usefulness of embedding machine-usable information into Internet documents, both commercially and for the individual user, will drive the implementation of these approaches.

1.4 Value-driven Information Gathering Applications

Numerous applications can benefit from a value-driven information gathering approach. There are two major criteria necessary for value-driven information gathering to be a viable approach: First, the system must have a decision model that can return the value of gathering a new piece of information, and second, the process of gathering information must be resource bounded. If these two conditions are met, then a value-driven approach may increase the overall effectiveness of the system compared to other approaches. Below is an examination of several potential value-driven information gathering applications:

1.4.1 Product selection

Making decisions based on information gathered from the Internet is the main experimental domain that this dissertation uses. Sites on the Internet contain a wealth of information about hundreds of products and services. They also have a fixed initial cost (often zero), take a variable amount of time to return results, sometimes do not return anything useful, and have varying accuracy levels. In the case of product selection, a person's individual tastes vary, but often their decision to purchase a product can be modeled by an expert based on several factors and then the decision model can be more finely tuned by the end-user through an interview process. This could involve ranking features that are most important or by filling out a survey that a system could use to determine the individual weights of specific features. Once the individual preferences of the users are discovered, the decision model can be modified without changing the actual structure of the decision model constructed by the expert.

The user also specifies the importance of time, money, and computational resources as they compare to finding the best product match. For example, a user might be willing to spend more money in order to receive a result in less time. These preferences are used to construct a cost function.

The modified decision model and cost function can then be used by the value-driven information gathering process to select a product that best matches the users individual requirements.

The value-driven information gathering system would search sites that include professional reviews, company announcements, individuals experience with the product, and other sources. Information sources in the context of the Internet do not have to be individual web pages or database, they could also be search engines or other web based experts. As long as the response-function and reliability of the information source can be accurately modeled, it can be incorporated into the value-driven information gathering system.

Three of the experimental systems described in Chapter 6 are implementations of the product selection systems using the Internet as the primary source of information.

1.4.2 Medical diagnostic systems

Medical diagnosis is an area in which excellent decision models exist, and statistical information about the responsiveness, cost and reliability of certain tests are well known. Often times in emergencies the decision of which tests are the most effective to run in order to make a treatment decision can save lives. Training a person to perform a number of tests is far easier to do then to give them a sense of which tests will be most effective in an emergency. Value-driven information gathering would work well in this situation because of the system's ability to deal with time-based probabilistic information sources. For example, a value-driven information gathering system could be created to help Emergency Medical Technicians (EMT). In this case, information sources would be different procedures that an EMT could perform in order to evaluate a patient's condition. The value-driven information gathering system would not only be able to determine which procedures would most effectively allow the system to come to a treatment decision, but the value-driven information gathering system could also use past experience with that individual EMT to more accurately evaluate the time required by that individual to perform the test.

1.4.3 Quality testing

Testing manufactured items to determine the potential for break down requires running a number of tests and correlating the result to make a decision on the probability that the item will fail. In this application, information sources are specific testing tasks accomplished by the device itself or other testing devices. For example, queries might include applying known input sets to the device and checking the results; or causing the device to fail at a specific point in operation and checking the internal state of the device; or examine the device with other testing devices. All of these testing actions are analogous to making queries of specific information sources and incorporating the results into a decision model. In this case, the action recommended by the decision model is to certify the device, reject the device, or recommend a specific repair to be done on the device.

The primary reason for describing this application is to stress the flexibility of the definition of an information source and a query. A query to an information source is a very flexible concept in value-driven information gathering. In these cases, a query is run on a device, whose internal mechanism the value-driven system knows nothing about. All that is important as far as the value-driven system is concerned is that the device has the ability to be queried and that the output of the device can be incorporated into the decision model used by the value-driven system. The responsiveness of the queries (in this case the queries are all actions performed by external pieces of hardware) is used primarily to allow the system to automatically determine the order in which specific test are run and to allow the value-driven system to provide a confident recommendation using the least amount of resources.

1.4.4 Personnel Management

Many organizations exist with the goal of gathering information and acting upon that information for a fixed problem. For example, local transportation departments are responsible for maintaining and inspecting bridges. The decision model for determining whether a bridge needs to be repaired as well as the tests that an inspection team can perform is known. The major challenge for the organization is to use its bridge inspection personnel efficiently. The goal is to inspect all of the state's bridges with a reasonable margin of safety while using the least amount of man-power.

Value-driven information gathering can be extended to model personnel as information gathering agents that can be assigned information gathering tasks and return information to the system at some point in the future, much like information sources. Using information gathering agents, a system can be developed that dynamically schedules and monitors personnel to maximize their effect on estimating the condition of each bridge.

The medical diagnostic system can also be expanded to use a large number of personnel that each have specific information gathering tasks which they can perform. Modeling a hospital as an environment of information sources and each patient as a decision model of what their current condition is would provide a mechanism for deploying and monitoring the personnel of an entire hospital and the patients that they are currently treating. With accurate decision models, a value-driven approach to personnel management could increase the quality of care by increasing the effectiveness of the personnel in the hospital.

1.5 Value-driven information gathering

Value-driven information gathering is a strategy for evaluating the effect that querying an external information source will have on the quality of a decision over time. This value (the *value of a query*) is based on the value of the information to the decision as well as the cost, reliability and responsiveness of the information source being queried. Previous work in decision theory defines half of this equation, and in many ways this work can be considered as an extension to decision theory that adds the cost of gathering information to the decision-theoretic notion of the value of information. Three key research questions are addressed by this dissertation:

1. *Combining decision theory with resource bounded reasoning.*
   Decision theory has developed methods for calculating the expected value of information to the process of making a decision. Resource-bounded reasoning has developed methods for determining the expected value of resource usage for a broad class of algorithms. Value-driven information gathering combines these two sets of techniques to develop a method for determining the value of information to a decision, when acquiring the information has a resource cost.
2. *Monitoring and planning information gathering in environments with a large number of varied information sources.*
   Previous work in information gathering on the Internet has focused on combining information from distinct information sources and the construction of plans that decompose complex queries into a set of atomic queries, which can be answered

by a set of sites on the web. In general, once this plan is constructed the task is complete. The execution of the plan is not examined. Value-driven information gathering selects the queries that will be made based on the results of the queries and when they return as the gathering and decision process are executing. This reactive approach allows the value-driven information gathering system to adapt based on the information that is returned by the queries as well as unexpected success of failure of queries to information sources[1].

3. *Developing methods for the creation of autonomous information gathering systems that use the Internet.*

   Currently, the number of complex software agents that operate in the domain of the Internet is still small (although the number is rapidly increasing as the web becomes more widely used). This thesis on Value-driven information gathering describes several software agents that operate on the Internet in non-trivial ways to answer complex questions using a large number of distinct Internet sources. It also describes an expandable system in which the value-driven approach can be used to increase the performance of many potential Internet applications.

1.5.1 Thesis

This dissertation presents a collection of algorithms that improve the performance of decision making systems in large-scale redundant information environments. *Performance*, in the context of this dissertation, is defined as the quality of the decision made by the system given a specific environment and the resources used to make that decision. The *quality of the decision* may be measured in two ways: Either by the accuracy of the decision (how often is the decision the correct one), or by the average utility of the decision (Assume that every decision the system could make is given a

[1] Unexpected success might seem like an odd occurrence, but in the context of value-driven information gathering it means a query that returns much more quickly then expected or with a much higher level of confidence then expected.

numeric score, what is the average score of the system compared to a system that always made the correct decision). The *resources used* are the time required by the system to return a decision as well as the money spent the system to query specific information sources.

A l*arge-scale redundant information environment* is a set of information sources, which the system may query in order to retrieve the information used by the system to make a decision. An *information source* is defined as any autonomous resource that has a probabilistic chance of returning information at some point in time after being queried. Latter in the dissertation we consider the cases in which the information returned by an information source is not always correct, in this case we refer to this information as *evidence*. An information source may also have a monetary cost associated with it, which must be paid by the system in order to query it. The information environment is *redundant* in the sense that for any specific datum of information, there might be several information sources that can return this datum. The term *large-scale* denotes that the system has access to so many information sources that querying all of them is not feasible given the resources available to the system making the decision.

This measure of performance is similar to the measure used by resource-bounded systems. Resource-bounded systems are generally evaluated by the quality of their results (usually measured by the error compared to the correct result) and the resources used to arrive at that quality of result. Because of this, resource-bounded systems that return results with large margins of error may still be considered effective, because they use very few resources and the environment they operate in is forgiving.

One goal of this work is to create a decision making system that we deem is *effective*. An effective decision making system is defined as having these three characteristics:

1. An effective decision making system must be able to return a decision given any amount of resources.
2. The average quality of the decision must smoothly degrade as the resources given to the system are reduced.
3. The system must be able to make accurate decisions regarding trading more resources for an expected increase in quality.

It is important to define this third characteristic in detail. Given a situation in which the system has a function that describes the utility of making a correct decision versus the utility of making an incorrect decision, and a function that defines the utility cost of using resources, an effective decision making system will be able to determine when it is worthwhile to continue gathering information (although it has a utility cost) and when to halt and return a decision.

An effective decision-making agent must accurately evaluate three aspects of any potential information source before deciding whether to query it. The three factors are used to calculate the *value of querying* a particular information source. Determining this value is a central focus of the dissertation and the value of a querying each available information source in the value-driven information gathering algorithm.

The first aspect for determining the value of querying an information source is to determine the value of the information to the decision, regardless of the resources needed to make the query. The value of this information varies based on the information the system already knows, the current set of outstanding queries, and the quality of the information source. This is equivalent to the *value of information* described in Pearl [58]. The second aspect that a decision-making system must consider in deciding whether to query an information source is the resources required to query the information source. The resource costs for the query might include the cost of time for waiting for the query to return, the monetary cost charged by the information source and the computational cost of processing the information (evidence) that is returned by the information source. The third aspect to consider is the resources that have already been spent gathering information prior to this query being made. These resources include the direct charge by the information source and the resources associated with waiting for the information source to return. The time that the decision-making agent takes to return a decision is crucial in evaluating the value of the decision. A high-quality decision will be useless if it takes too long or cost too much for the system to make it.

The improvement in overall decision quality using a value-driven information gathering approach in a restricted resource environment is a result of several factors.

Proving that these hypotheses are true in a rich information source environment is another contribution made by this dissertation.

It is not feasible to gather all the information associated with a decision

In large information environments, the process of gathering all of the available information would take an unacceptable amount of time and monetary resources.

It is advantageous to halt information gathering well before all of the information sources have been queried

In general, the marginal value for each subsequent query decreases as more queries are returned. The cost function, on the other hand, is always increasing. This implies that there is a point at which halting the information gathering process and returning a result, even one with a lower decision value is a more optimal strategy than to continue the information gathering process.

The computational resources required to extract evidence from information sources will increase in the future

As more sophisticated information extraction algorithms are developed (for example, natural language processing), they will become more automated and be able to extract information from an increasingly broad class of information sources on the internet. They will also become more computationally expensive, further increasing the fixed computational cost as well as the time required to collect information from information sources.

Spending computational resources to determine which subset of information sources to query will increase the overall quality of the decisions made by the system

As the number of potential information sources increases and the resources required for extraction increases, the value of spending computational resources during the information gathering process to select the most beneficial information sources will also increase.

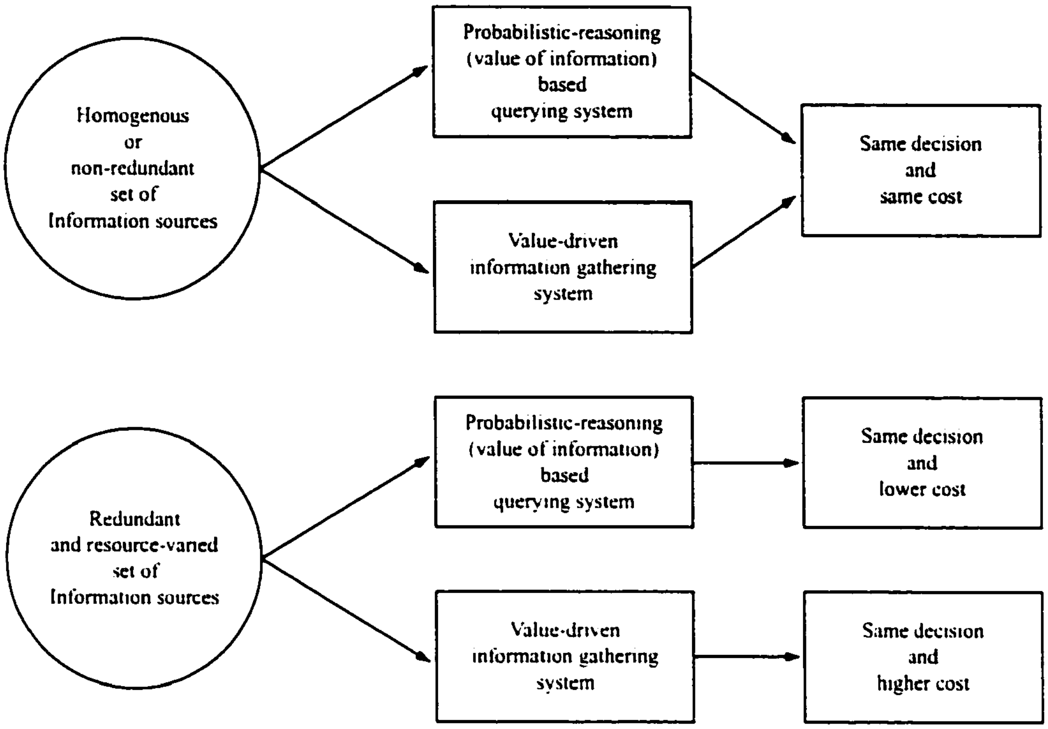


Figure 1.1 – Value-driven information gathering approach versus an information gathering system based only of the value of information

It is beneficial to use a value-drive approach

Considering that a value-driven approach takes into account both the quality of the information as well as the resource cost for making the query, a value-driven approach will perform significantly better in a restricted resource environment then any previous approach.

A value-driven system can be easily modified to construct a broad class of decision support systems

The final system presented in this dissertation represents a good framework for the creation of a large number of decision making agents. In order to create a decision making system for a new domain, the user only needs to specify an influence diagram for the decision, the set of information sources, and the set of information extractors.

The goal of this dissertation has been to construct a system, which facilitates the creation of web-based resource conscious decision making systems. These systems can both autonomously make decisions or assist individuals and organizations with complex decisions with resource restrictions. This dissertation demonstrates that it is possible to develop algorithms that reason about the potential benefit of querying an information source, and that these algorithms drastically improve the performance of the decision making system.

A value-driven information gathering approach is the best solution to this problem because it leverages the large body of research done in probabilistic reasoning with the research that has been done in resource-bounded reasoning. The computational cost of determining the value of a query is very close to the computational cost of determining the value of information. By reasoning about both the information benefit and the query costs, a value-driven system can query a set of information sources that will use fewer resources while returning a decision of the same quality as a system that only used the value of information. The net effect is a decision of the same quality that requires fewer resources in domains with a large number of redundant resources, and a decision of the same quality using the same resources in domains with fewer information sources or a set of homogenous information sources. Using a value-driven approach will not result in a lower-quality decision, in unfavorable domains the decision quality will be the same as a simpler approach.

1.6 Outline

Chapter 2 describes previous work that has lead to the development of value-driven information gathering. This section includes background material in decision theory, information gathering, and resource-bounded reasoning. The chapter also compares current work in these fields to value-driven information gathering.

Chapter 3 formally defines the problem of value-driven information gathering.

Chapter 4 describes the process of value-driven information gathering. First, the motivation and algorithm are described and then the mathematical methods for calculating the value of a query are explained.

Chapter 5 describes in detail the components of a value-driven information gathering system. This chapter also discusses the messaging system used by each component.

Chapter 6 describes the five experimental systems that were developed over the past three years for value-driven information gathering. Each section of this chapter describes the system, the value-driven components it uses, the environment it operates it, the tests that were done, and the results produced by the system. The final section includes a trace of a demonstration gathering session.

Chapter 7 summarizes the results of the experimental systems and explains the contributions of the research. Future work that would expand the capabilities of value-driven information gathering is also discussed.

CHAPTER 2

BACKGROUND

The algorithms and systems presented in this thesis are based on previous works in three areas of computer science. *Decision theory*, which concerns algorithms that maximize expected utility in probabilistic networks with uncertain information. *Resource-bounded reasoning*, which deals with the construction of meta-level algorithms that balance the resource cost of progressive computation with the benefit of improving the quality of the result. And *information extraction*, which deals with the construction of systems for extracting and combining complex data from disparate database and services.

Value-driven information gathering contributes to all three of these areas of computer science. In decision theory, this thesis introduces the *value of a query*, which expands on the value of information to include a time dependent cost for instantiating a variable node in the influence diagram. In resource bounded reasoning, value-driven information gathering presents an algorithm to evaluate the optimal point at which to halt a resource restricted operation, querying external information sources. In the field of information extraction, value-driven information gathering presents a mathematically based method for combining redundant (and possibly conflicting) information from separate databases that take the accuracy, bias and range of the database into account.

2.1 Decision Models for Reasoning under Uncertainty

Decision theory is an area of research that focuses on the propagation of information through a network of causally connected nodes. These nodes are called variable nodes, because they represent a random variable in the world. The network is a directed graph, and in a majority of cases, the network is also acyclic. This network is called a *Belief* or *Bayesian* network. Belief networks were developed in the late 30's to the early 50's with work by de Finetti [16], Good [32] and Savage [66]. **Figure 2.1** shows two variable nodes in a belief network. **Figure 2.1** also shows the casual connections that exist between each of the three variable nodes. Nodes without any arcs leading from them are

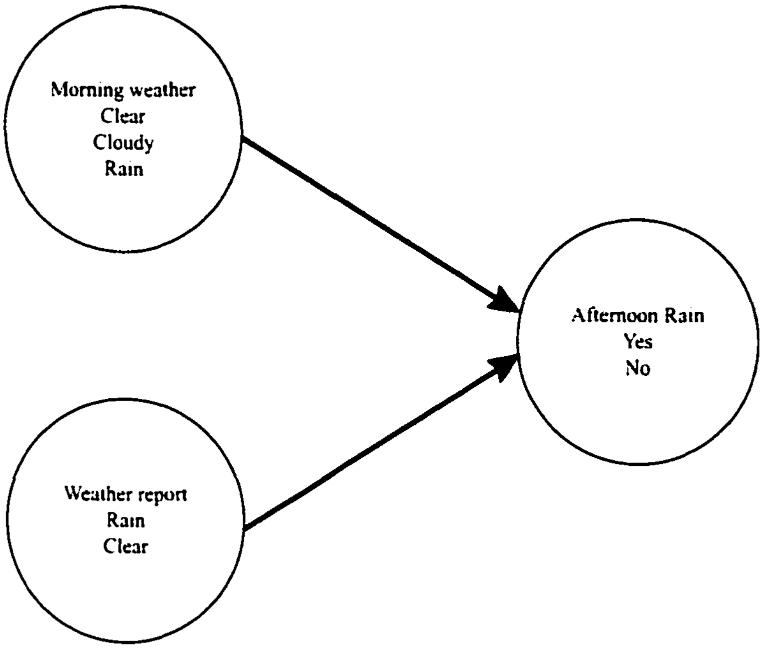


Figure 2.1 - A simple belief network with three variable nodes

called leaf nodes, and nodes with arcs exiting from them are called internal nodes. Each node has a finite number of states and a node may be either instantiated or uninstantiated. An instantiated node is a node whose state we know with absolute certainty. A uninstantiated node is a node whose state is unknown. Both leaf and internal nodes may be instantiated or uninstantiated.

Definition 1 A variable node x_i has a set of states $s_{i,1} \ldots s_{i,k_i}$ with a probability distribution $\Pr(x_i = s_{i,j} \mid \mathit{Parents}(x_i))$ where $\mathit{Parents}(x_i)$ are the parent nodes of the variable node in question and $\sum_{j=1 \ldots k_i} \Pr(x_i = s_{i,j} \mid \mathit{Parents}(x_i)) = 1$.

Definition 2 An uninstantiated node is a node in which the state of the node is unknown. In such a case, the probability distribution of the variable node is calculated using the conditional probability table.

States of input node *Morning weather*	**States of input node** *Weather report*	**Output state of** *Afternoon rain*	
		Yes	**No**
Clear	Rain	0.7	0.3
Clear	Clear	0.3	0.7
Cloudy	Rain	0.8	0.2
Cloudy	Clear	0.6	0.4
Rain	Rain	0.9	0.1
Rain	Clear	0.7	0.3

Table 2.1 – Conditional probability table for the node *Afternoon rain* (see **Figure 2.1**)

Definition 3 The conditional probability table is a table containing every combination of states for the parent nodes of the variable node and a probability distribution for the states of the variable node (see **Table 2.1**).

Definition 4 An instantiated node is a node in which the state of the variable node is known to be one of the states in $s_{i,1} \dots s_{i,k_i}$.

Each node in the network can be represented by a table of real values whose width is equal to number of states the node may pass on to its children node and whose height is equal to the total number of combinations for the states of its input nodes. **Table 2.1** shows the conditional probability table for the node *Afternoon rain.*

Nodes that have no input connections are also represented by a probability table, but with only one row of values and no input states.

Belief networks are capable of describing a large class of casually connected states of the world. One advantage is that the belief network representation takes advantage of independent relationships that exists between nodes in the network. Instead of one

Output state of *Morning weather*		
Clear	**Cloudy**	**Rain**
0.3	0.5	0.2

Table 2.2 – Probability table for *Morning weather* (see **Figure 2.1**)

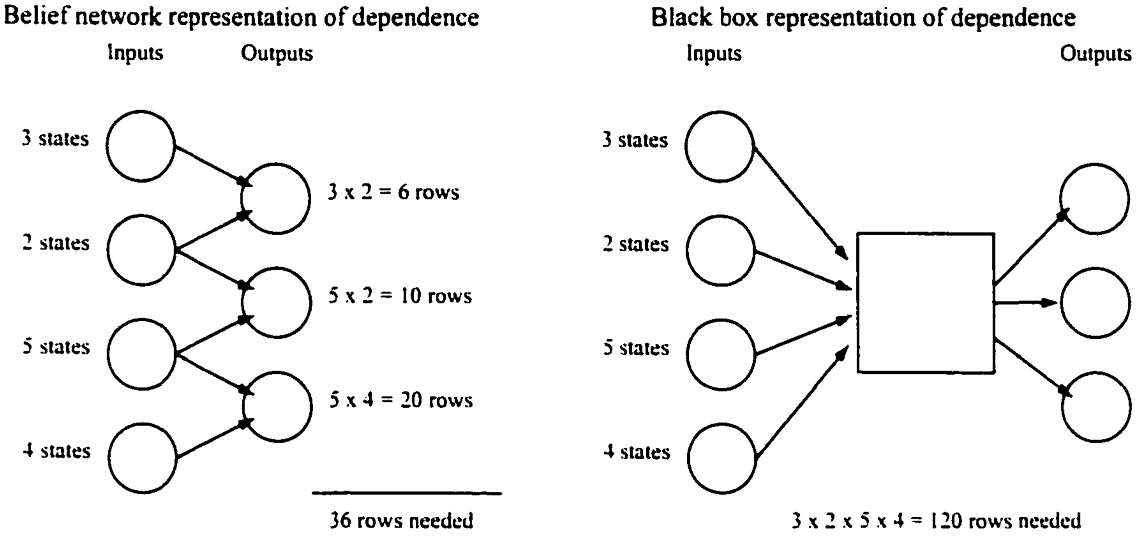


Figure 2.2 – Data required to represent the input and output states of a probabilistic network using the belief network approach and a black box approach.

gigantic table, which is has as many rows as all possible states of the input nodes, a belief network has a table for each node in the belief network with as many rows as the number of combinations for the nodes that directly influence it. **Figure 2.2** shows the difference in the amount of data required to represent a four input node, three output node network using a belief network representation and a black-box approach.

The belief network representation contains information connecting each output node to the nodes that influence their value. Therefore, the top-most output node only needs a table with six rows of data to represent all of its potential output values. Six rows represent all of the possible combinations of values for the two input nodes that influence the value of the output node. For the black-box representation, there is no information about the dependence between nodes and thus in order to calculate the top-most node (or any of the output nodes value) we must look the value up in a table that is 120 rows in size. All 120 rows are required because we need to represent every possible combination of input nodes (3 x 2 x 5 x 4 = 120). In networks that are more complex it would be infeasible to store a table large enough to represent every possible combination of values for the input nodes.

The primary disadvantage to using a belief network is the case in which we have chains of variables nodes. The system cannot instantly return the results of the output nodes using look-up. Instead, the output node value must be determined by propagating the input value through the conditional probability tables in the network. The algorithm for determining the probability distribution for a node given a network and a set of nodes that are instantiated to specific values is not complex, but performing this calculation can be computationally expensive for large networks. The complexity for propagating a belief network is linear in the number of nodes for acyclic graphs and poly-trees, and NP-hard for directed acyclic graphs (DAGs). Although the number of states that a belief network may have is finite, after propagating the network, each variable node has a probability distribution for the set of states. For example, after instantiating the two nodes *Morning weather* and *weather report* and propagating the network, the node *Afternoon rain* might have a probability distribution of: Yes = 0.6 No = 0.4.

The networks used in decision theory can generally be divided into two classes: belief

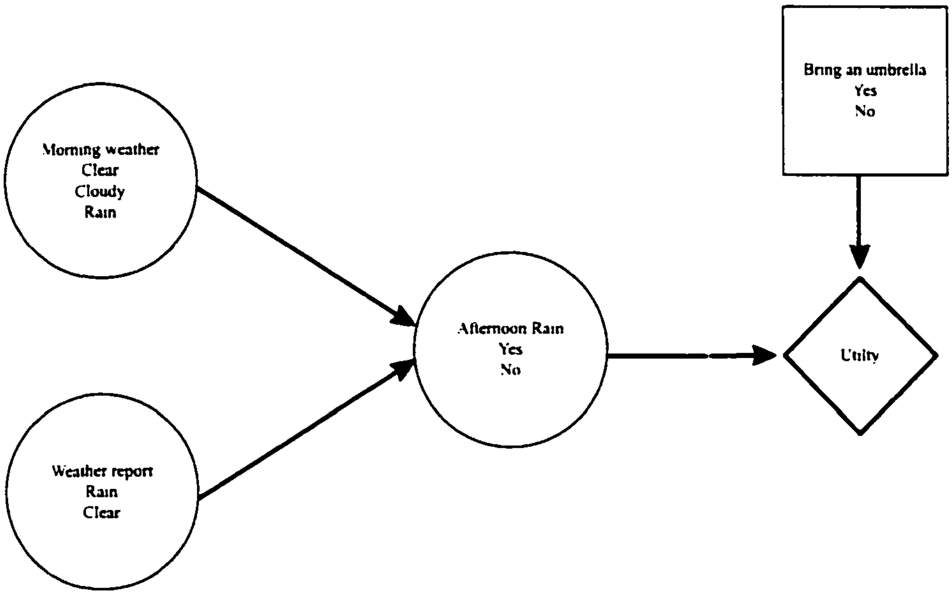


Figure 2.3 – An influence diagram constructed from the simple three node belief network (see **Figure 2.1**)

States of input node *Afternoon rain*	States of decision node *Bring umbrella*	Utility
No	No	0
No	Yes	-1
Yes	No	-3
Yes	Yes	2

Table 2.3 – The utility table for the afternoon rain influence diagram (see **Figure 2.3**)

networks (described above) and influence diagrams. Belief networks contain only variable nodes. Influence diagrams contain variable nodes and two other types of nodes: Utility nodes and decision nodes [47]. Influence diagrams are used both to evaluate the probability distribution of a set of variable nodes in the network and to return a decision that will maximize the expected utility of the system. **Figure 2.3** shows the "afternoon rain" belief network (see **Figure 2.1**) expanded into an influence diagram by attaching a utility node and a decision node.

A *utility node* is a leaf node that uses the probability distribution of a set of nodes (variable and decision) to return a specific utility score. The utility node is represented by a matrix of utility values based on state of the input nodes, much as the conditional probability table described above (see **Table 2.3**).

Since the input nodes usually have a probability distribution over a set of states, the value for the utility node is calculated by summing the utility score for every combination of input node states multiplied by the probability of that combination of input node states being true.

$$Utility = \sum_{\alpha \in I} U(\alpha)\Pr(\alpha) \qquad (2.1)$$

Where α is a set of input states from the set of all possible input states values I.

$I = \{\{s_{1,1},\cdots,s_{n,1}\},\cdots,\{s_{1,k_1},\cdots,s_{n,k_n}\}\}$

$s_{i,j}$ is the state i for the input node j

n is the number of input nodes for the utility function

k_j is the number of states for input node j

A *decision node* also has a finite set of states. Each state represents a possible decision the system can make. The decision node must be instantiated with a particular decision when the influence diagram is evaluated. The influence diagram evaluation algorithm determines which decision maximizes the value of the utility node for the influence diagram.

The influence diagram can be evaluated to determine which decision will maximize the expected utility of the system given the probability distribution of the variable nodes in the belief network.

Influence diagrams that are used in real-world applications have a much greater number of variable nodes to represent much richer domains. Even in the umbrella decision, we could include variable nodes based on the weather the day before, the time of the year, or other forecasts. In most cases, as more variable nodes are instantiated with information, the variance in the expected utility of the decision decreases. This increase the confidence the system has in the utility maximizing decision.

Influence diagrams were developed from work in decision theory, probabilistic reasoning, and game theory. They have been used in a wide variety of fields, including economics, medicine, statistics, and computer science. Influence diagrams were introduced by Howard [44][45] and algorithms for evaluating them are discussed in detail by Pearl [58], Shachter [68] and Castillo [12].

Decision theory emerged in the 1940's from work by von Neumann and Morgenstern [71] in game theory. Game theory initially focused on choosing an optimal move to make in deterministic games with perfect information (e.g. chess), but these restrictions were relaxed to include games in which the players did not know the full state of the world and that also involved randomness (e.g. poker). At this point game theory was powerful enough to address many real-world problems. Game theory became the basis for much of modern economics and the foundation for decision theory in computer science.

The goal of game theory is to evaluate the state of a game and select a move or action that maximizes the expected payoff. Initially game theory used *decision trees* to determine the course of action that would maximize the payoff of a particular game. Game theory was eventually expanded to include incomplete knowledge, but game theory did not initially incorporate probability theory to represent random variables and their causal relationships.

Game theory also had an absolute scale for rating each possible resolution. In a simple win/lose game, the value of winning the game would be 1 and losing would be -1. In more games with multiple outcomes (such as blackjack or poker), the score given to various outcomes might be based on the amount paid off for a one dollar bet. One of the major contributions of early decision theory was to develop a mathematical system for evaluating actions on one universal scale, *utility*.

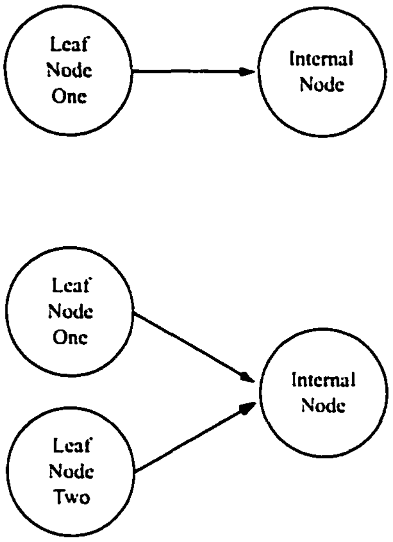
Decision theory allows the utility of performing an action in a domain to be evaluated in a meaningful way. This paradigm, of evaluating all potential actions on one scale and executing the action with the highest expected payoff is widely used in artificial intelligence systems today. The challenge is to develop accurate representations of the world and good *evaluator functions* that will return accurate utility values for any state in the domain. The construction of good evaluator functions that map world states to accurate utility values is a difficult task that not only requires expertise in the field, but a strong understanding of the effects of the evaluator function on the behavior of the system. Many researchers are currently working on constructing high-quality evaluator functions and belief networks for specific domains. There is also a large amount of research being done in automating this task through various learning algorithms.

Another use for influence diagrams is calculating the *value of information* for any variable node in the influence diagram. The value of information is the increase in expected utility of the system if a specific variable node were instantiated with the correct value. The value of information can apply both to one variable node and to a set of variable nodes. Howard [44][45] developed methods for calculating the value of information by comparing decision quality of the influence diagram before and after a set of variable nodes in the influence diagram have become instantiated. Calculating the value of information for a set of variable nodes is one of the steps in determining the *value of a query* used in value-driven information gathering.

Calculating the value of information is extremely useful for a large number of real-world applications. In the medical domain, it can be used to determine which medical test to perform, in equipment testing it can be used to determine which test-bed to run next. Unfortunately, it is an extremely expensive computational operation to perform. Evaluating an influence diagram is a NP-hard computation. The belief network must be evaluated once for every combination of states contained in the set of nodes. Based on the topology of the influence diagram, each evaluation of the influence diagram may take linear time (for an acyclic influence diagram) or may itself be an NP-hard problem (for directed acyclic graphs). This high computational cost has lead to the creation of a set of anytime algorithms that address the problem. Horvitz [38][41][39] has worked on the problem of adding resource-bounded constraints to the evaluation of influence diagrams. This research has used influence diagrams themselves as a mechanism for controlling computation in the decision making process. Horvitz has focused on influence diagrams used in medicine, a domain in which fully solving the large influence diagrams is often not feasible with the time constraints inherent in the domain. Horvitz has also extended this work to situations in which the resource constraints are uncertain and change as the system is executing.

2.1.1 Incorporating evidence into belief networks

In many cases, variable nodes in a belief network cannot be instantiated directly by the value returned by an information source (see **Definition 4**) because of uncertainty about the information received by the system or because there are multiple sources of information for the same variable node. Part of the work in value-driven information gathering has been to create an automated mechanism to deal with these situations. Belief networks do not have a mechanism for dynamically adding new nodes to the network and connecting them to a pre-existing variable node (see **Figure 2.4**). The problem is that the conditional probability table that the belief network propagation algorithm uses is dependent on every possible combination of values contained by the parent nodes. When we add a new leaf node to the network, the effect of the new node on the probability distribution of the internal node is based on the states of all of the pre-existing leaf-nodes that connect to the internal node. The pre-existing leaf nodes also must take all of the states of the new leaf-node into account as well. For example, in



Conditional probability table

Leaf Node One	Internal node True	Internal node False
True	0.4	0.6
False	0.2	0.8

Conditional probability table

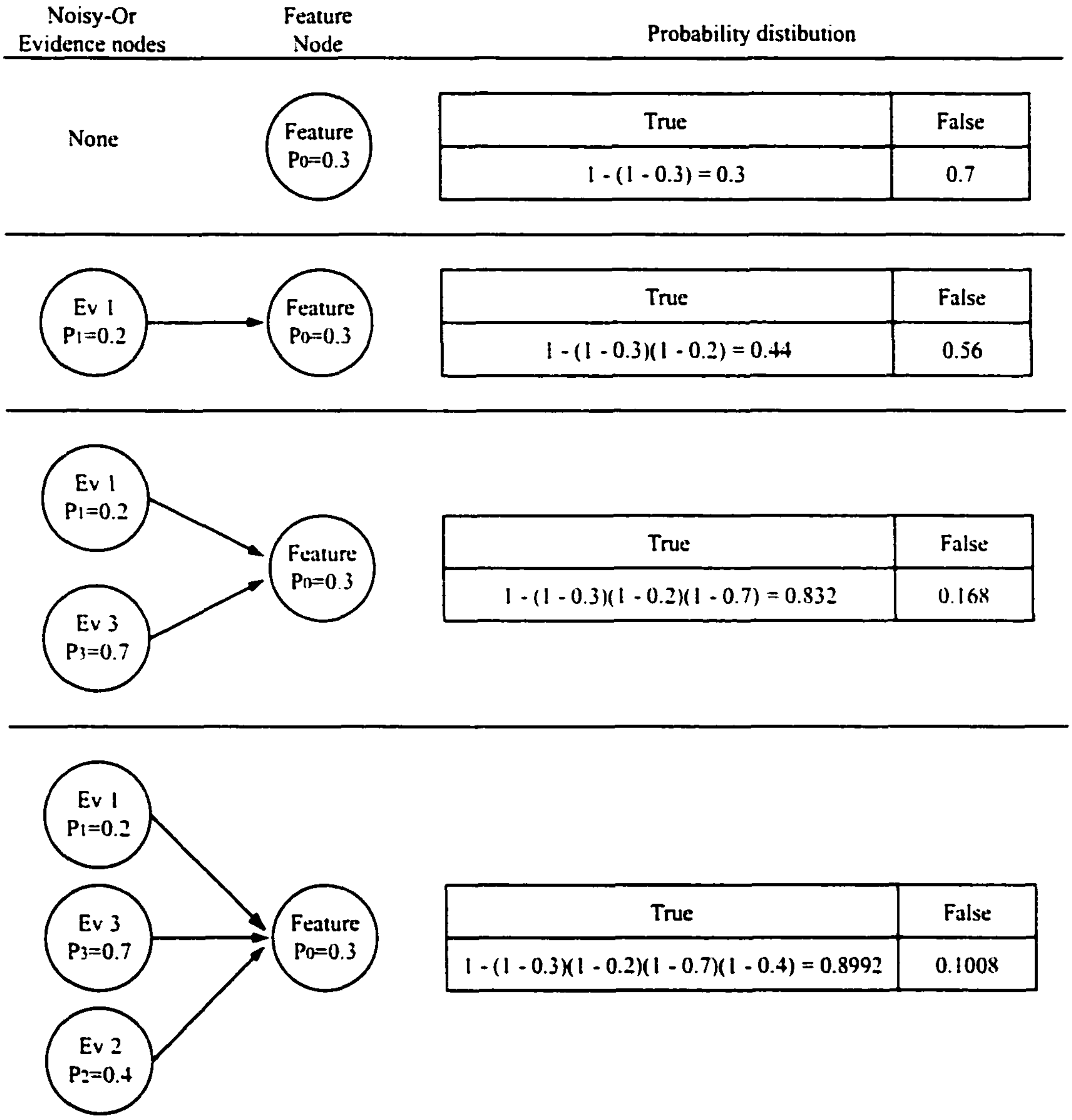
Leaf Node One	Leaf Node Two	Internal node True	Internal node False
True	True	0.1	0.9
True	False	0.3	0.7
False	True	0.4	0.6
False	False	0.5	0.5

Figure 2.4 – Adding a new node to a Belief Network

Figure 2.4, adding leaf node two forces the system to generate rows for every combination of input states. The system must know how leaf node two being true or false effects the influence of leaf-node one.

In order to be able to incorporate evidence dynamically into a belief network, a system must be developed that can dynamically generate conditional probability tables for the new nodes that will be dynamically added to the belief network. A conditional probability table can describe situations in which the effect of parent node A and parent node B can have a large synergistic effect or can cancel each other out. A system that creates conditional probability tables must make assumptions about the relationships between the existing pieces of evidence and any new piece of evidence that needs to be incorporated into the belief network.

Value-driven information gathering use a system for dynamically creating and connecting a restricted class of variable node to the belief network as new evidence is incorporated into the decision model. These restricted nodes are called *Evidence nodes* and are described in more detail in section 3.1.3. Evidence nodes also have the advantage of using the same belief network propagation algorithm as normal variable nodes.



True	False
1 - (1 - 0.3) = 0.3	0.7
1 - (1 - 0.3)(1 - 0.2) = 0.44	0.56
1 - (1 - 0.3)(1 - 0.2)(1 - 0.7) = 0.832	0.168
1 - (1 - 0.3)(1 - 0.2)(1 - 0.7)(1 - 0.4) = 0.8992	0.1008

Figure 2.5 – A Noisy-Or node

Other approaches for incorporating evidence into a decision model have been to create three new types of evidence nodes: *Noisy-OR*, *Leaky Noisy-OR* and *Noisy-Max* nodes [37][60]. All three of these node types allow a decision model to incorporate new pieces of evidence without having any knowledge of the effect of particular sets of evidence on other pieces of evidence. This is done by making specific implicit assumptions about the dependency between new pieces of evidence that the system incorporates into the model. In this case, the assumption is that evidence only provides support for a specific state in the belief network. It is not possible to add evidence that decreases the probability of the variable node being true (in the case of *Noisy-OR* and *Leaky Noisy-OR*) or having a lower value (in the case of *Noisy-Max*). Often, this is the dependency model that the designer of the network wishes to use, but it is important to understand the restrictions placed on the type of evidence that may be added to these systems.

Figure 2.5 shows a Noisy-Or node in a Belief Network. A Noisy-Or node has only two states: True and False. Noisy-OR nodes require one piece of meta-information for each potential piece of evidence. This additional value is called the *level of support*. The level of support is the probability that the feature x is true when only information source s_i is true.

$$ps_i = \Pr(x \mid s_i \bigwedge_{i \neq k} \overline{s_k}) \tag{2.2}$$

If the system knows the level of support for each piece of evidence, it is possible to determine the probability that a feature x is true given any set of evidence. Using equation (2.2) we can derive the probability value for x given a set of supporting evidence nodes that have been found and their level of support.

$$\Pr(\bar{x} \mid E) = \prod_{s_i \in E} \Pr(\overline{s_i})$$

$$\Pr(\bar{x} \mid E) = \prod_{s_i \in E} (1 - ps_i)$$

$$\Pr(x \mid E) = 1 - \prod_{s_i \in E} (1 - ps_i) \tag{2.3}$$

Where x is the variable represented by the Noisy-Or node

E is the set of supporting evidence (s_i) that has been found

ps_i is the level of support for each piece of evidence

Noisy-Or nodes can be extended to include a base-line probability that the node is true, even when no supporting evidence has been found. These nodes are called Leaky Noisy-Or nodes and include a constant ps_0 called the *leak value*. The probability of a Leaky

Noisy-Or node being true is described in equation (2.4). **Figure 2.5** shows the Noisy-OR computation for a set evidence nodes and the feature node, which they lend support.

$$\Pr(x \mid E) = 1 - (1 - ps_0)\prod_{s_i \in E}(1 - ps_i) \qquad (2.4)$$

Where x is the variable represented by the Noisy-Or node

E is the set of supporting evidence (s_i) that has been found

ps_i is the level of support for each piece of evidence

ps_0 is the leak value for the node

Noisy-Max nodes are very similar to Noisy-Or nodes with the exception that they allow for more than two states and that these states must be arranged in order. For example, a Noisy-Max node might contain the states low, medium and high. As new evidence becomes available the probability distribution for the Noisy-Max node shifts to the maximum value of the evidence nodes.

Noisy-Or and Noisy-Max nodes are limited in the type of evidence that they can represent. They can only represent the presence of evidence in the network and not any information about the state of the evidence. In addition, these node types are only additive in nature, new information can add support to a feature being in a particular state, but cannot remove support. They cannot be used to represent counter-evidence. Noisy-Max nodes are able to have multiple values for the variable they represent, but evidence used by the Noisy-Max nodes only provide evidence that the variable has at least that value. No piece of evidence can decrease the probability distribution of the Noisy-Max node.

2.2 Resource-Bounded Reasoning

Resource-bounded reasoning is a field that involves several research directions and many techniques for developing algorithms that consider computational resources. Resource-bounded systems were first described by Herbert Simon in economics. These

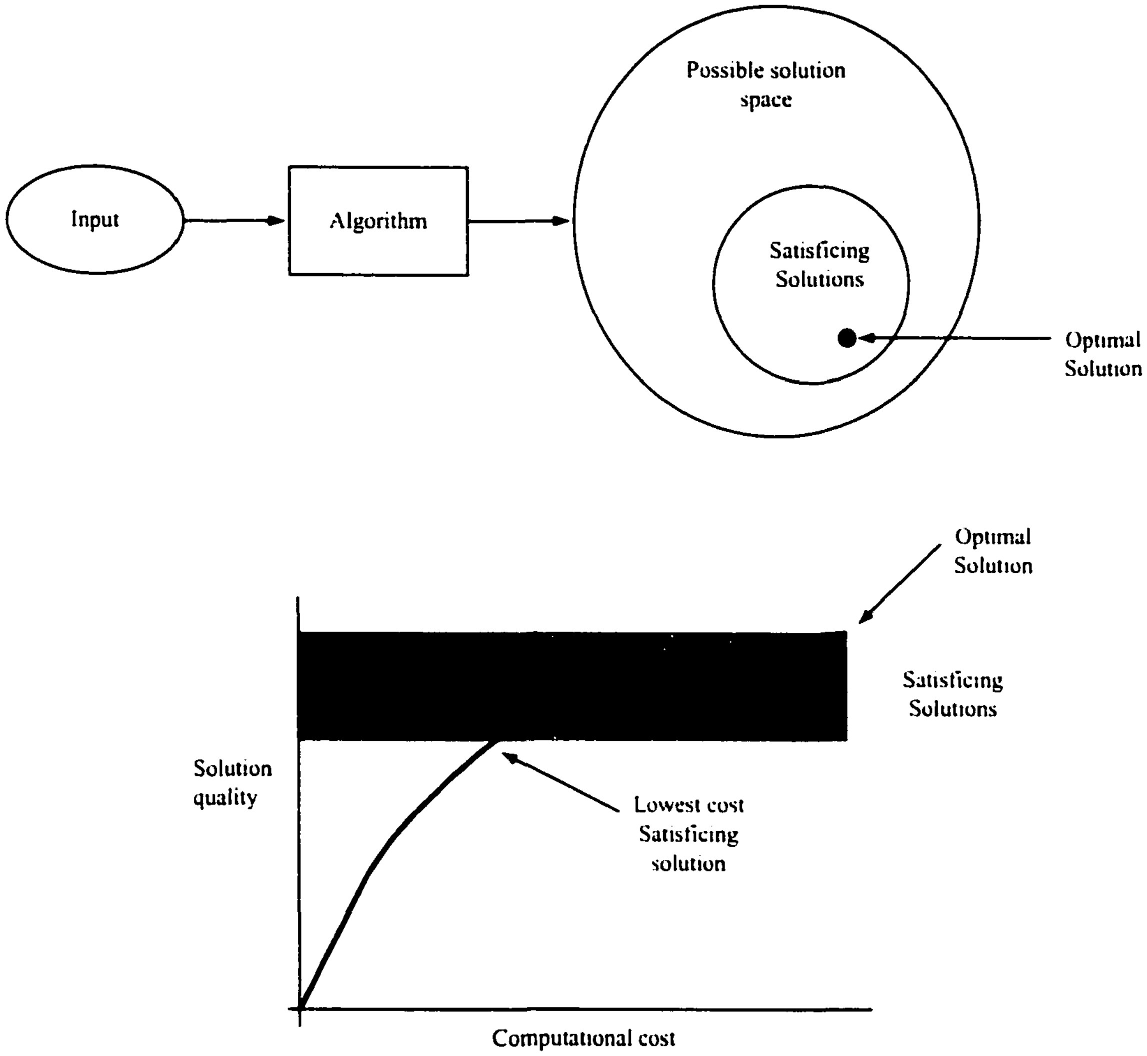


Figure 2.6 – Algorithms with a satisficing set of solutions

systems (which could be individuals or firms) were described as having *bounded rationality*. Simon also defined the term *satisficing*, which is defined as a solution that, while not optimal, meets the minimum requirements of the system and uses fewer resources to compute than an optimal solution. I.J. Good divided all decision making systems into either *type 1* or *type 2*. Type 1 systems strive to reach optimal decisions regardless of the resources used. Type 2 systems consider the resource requirements of making a decision and reach a solution that is satisficing.

Simon explained the strategy behind bounded rationality in three ways: First, in the real world a great deal of the information that is used in making a decision is changing over time; if a system spends too much time making a decision, the information used to make that decision will be out of date. Second, the rate of improvement for most algorithms is decreasing. The greatest improvements in the quality of the algorithm will occur early in the process and additional time spent on the computation will have a smaller and smaller effect. Three, decision making is not an isolated task performed by the system; there are other tasks the system can be doing to improve its overall performance of the system and spending too much time on any individual decision will begin to interfere with the other decisions or monitoring tasks that the system needs to perform. When analyzed in this way, systems with bounded rationality are still making optimal decisions, but the equation that determines the value of a decision now includes the resources used in the decision as well as the resources the system has to spend.

The rationality of human behavior has often been used for the development of techniques used in computer science [19]. The application of bounded rationality and meta-reasoning in artificial intelligence has allowed the creation of systems that can operate in time-restricted environments, which traditional AI systems would not be fast enough to operate in [43][63][64][76].

A large number of problems in artificial intelligence exhibit the two qualities that make meta-reasoning a powerful approach: Often times there is a large area in the solution space that is satisfactory for the system, and computing the optimal solution will use an infeasible amount of resources. Many problems in planning and search have execution times that are related exponentially to the size of the problem.

One example of a successful application of resource bounded reasoning in artificial intelligence is the work by Eric Horvitz in medical diagnosis using influence diagrams [43]. The Protos system uses an influence diagram to evaluate patients suffering from respiratory problems. Influence diagrams are an extremely effective tool for successfully diagnosing patients in the domain of medical diagnosis. The primary problem with the use of influence diagrams for the diagnosis of patients is that it is extremely computationally expensive to evaluate an influence diagram of the size and connectivity used in medical diagnosis. In many instances it would be impossible to fully evaluate these influence diagrams in real-time. Horvitz developed *bounded conditioning*, which is an extension on Pearl's evaluation algorithm that allows the nodes in an influence diagram to be incrementally refined. For each node, a cut set is created from the immediate parents and children of the evaluation node (see **Figure 2.7**).

Once the probability distribution for each node in the cut set is determined, the combined probability for each possible combination of nodes is calculated and sorted. To determine the probability of the evaluation node, the algorithm instantiates the set of nodes with their corresponding values and determines the probability distribution of the evaluation node. The result is multiplied by the combined probability for the values of the cut set and added to the total. This gives an approximation of the probability distribution for the evaluation node as well as a range of possible probabilities for each value.

The optimal decision selected by the influence diagram can be determined at any time while the quality of the probability distribution for each of the nodes is being improved. By determining the effect on the decision of refining the node probability distributions, it

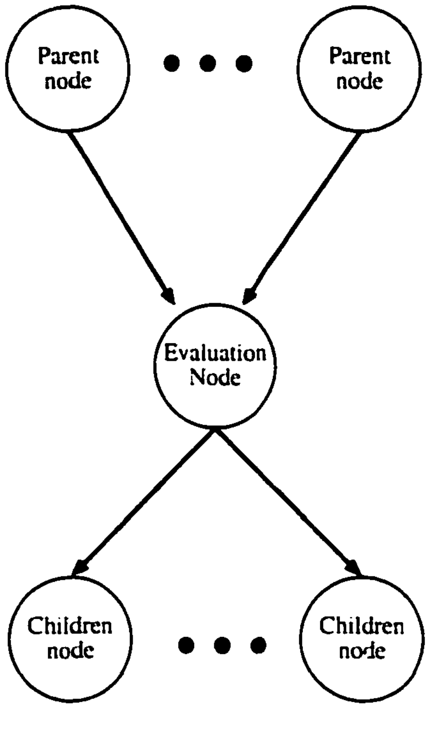


Figure 2.7 - The cut set for any node in an acyclic belief network

Input states	Probability
$s_1 = \{v_{1,k_{1,1}}, v_{2,k_{1,2}}, \ldots, v_{n,k_{1,n}}\}$	pr_1
$s_2 = \{v_{1,k_{2,1}}, v_{2,k_{2,2}}, \ldots, v_{n,k_{2,n}}\}$	pr_2
$\vdots$	$\vdots$
$s_m = \{v_{1,k_{m,1}}, v_{2,k_{m,2}}, \ldots, v_{n,k_{m,n}}\}$	pr_m

Where s_x is the set of input states for row x

$v_{i,j}$ is state j for input node i

$k_{x,i}$ is state index for row x for input node i

pr_x is probability of the set of states for row x ($pr_x < pr_{x-1}$)

Table 2.4 - The probability distribution for each combination of the cut set

is possible to calculate the expected value of computation (EVC). The expected value of computation returns the expected improvement in the expected utility of the influence diagram for refining the probability distributions of the nodes. This value is represented by the following equation:

$$EVC(\vec{r}') = \sum_{\vec{v}} u_c(\vec{v}',\vec{r}')\Pr(\vec{v}' \mid \vec{r}') - u_c(\vec{v},\vec{r}) \tag{2.5}$$

Where $\vec{r}'$ is the new refinement

$\vec{v}'$ is the new probability distribution of the variable node

$u_c(\vec{v}',\vec{r}')$ is the new expected utility of the system given the new probability distribution of the variable node

$\Pr(\vec{v}' \mid \vec{r}')$ is the probability or the new distribution ($\vec{v}'$) given the new refinement ($\vec{r}'$)

$u_c(\vec{v},\vec{r})$ is the expected utility of the system given the old probability distribution of the variable node

The EVC represents the expected increase in utility of the network (calculated by making the decision with the highest expected utility) if a new refinement is incorporated into the influence diagram. Refining the probability distribution of the node is defined as calculating the new probability distribution for the node by including the next row in the state probability table (see **Table 2.4**). As the refinement algorithm processes each row in the state probability table, the influence that this combination of states has on the decision decreases. The decrease in influence comes from two factors: First, because the probability of this combination of states is decreasing as we descend, this decreases the amount that the probability distribution can change. Second, there is a high likelihood that after evaluating a few rows in the table, the probability distribution will reach a state where it is no longer possible for the decision to be altered, regardless of the effect of the rest of the rows.

Using the EVC, the Protos system can determine when to halt computation and return a result when either of two events occurs: The influence diagram is refined enough that no refinement will change the decision, or the resources (time) spent refining the influence diagram are outweighed by the additional risk to the patient brought about by the delay.

Anytime algorithms [9][17][75][76] have a constantly improving output, much like iterative improvement algorithms. They also are able to make predictions about the expected quality of the output at any given time after activation. The expected quality of the output given a set of resources and the quality of the input is represented by a *performance profile*. The utility of the computation is calculated using the quality of the output and the cost of the resources used. Using the performance profile and the cost function, an anytime algorithm can return results with maximum utility. Performance can be increased even more with the adoption of sophisticated monitoring schemes that adapt the time allocation based on the current output quality [36].

Flexible computation systems control the time required by an algorithm by controlling the computation resources given to the various algorithms used by the process. In the case of an influence diagram, the resources used by the algorithm are controlled by allocating resources to the Belief propagation algorithm. The Protos system [38][39][42][40][43] has applied this method to several time intensive problems in the medical domain. Influence diagrams in the medical domain are generally large enough that completely propagating the network would take too much time. By reasoning about the effect of the belief network propagation on the decision, the system can determine at which point computational resources will no longer alter the decision.

Design-to-time algorithms [27] compose plans out of several components in order to arrive at a high-quality decision given a set amount of resources to use. This work has been expanded to design-to-criteria planning [74] in which both the resources used and the numerous qualities of the output are considered (see 2.4.3 for more details on design to time and the BIG system).

Decision-theoretic meta-reasoning [61] uses information value theory to determine which functions to use in order to maximize the output quality given a set of resource constraints. By evaluating the current set of data and the operations that may be performed on it, more effective methods for directing the algorithm can be developed.

Value-driven information gathering uses many resource-bounded reasoning techniques in an effort to optimize the quality of the decision given a set of resources. This allows a value-driven system to exhibit the properties of an anytime system. Value-driven information gathering systems return a result at any time and can develop a strategy that optimizes resource use if the resource restrictions are given ahead of time.

2.3 Information Extraction & Integration

Value-driven information gathering accesses a large number of different types of information sources. One of the key problems in gathering this information is extracting the data that can be used in making the decision from semi-structured documents. The problem of information extraction and integration can trace its roots to natural language processing and the construction of semantic networks. Both approaches allow systems to extract machine usable information and make inferences using rules generated from the text. Currently, a large number of people are developing systems that automate the process of extracting and integrating information from sources on the Internet. Before the internet became such an active research domain, a large number of researchers were developing the concept of digital libraries. Research on digital libraries has focused on four key problems associated with collecting relevant information from a large and diverse domain of information sources [8][26][31][30][46]: Locating relevant material, interacting with the individual information sources, extracting information from those sources and integrating the material.

The digital library system developed at the University of Michigan [8] had a simple planner that used a collection of simple and specialized systems to extract data from different information sources and correlate the results with a message passing architecture between the specialized systems. Each system would be capable of working on one simple task. Some systems could find relevant information, others could communicate with specific systems that the digital library accessed, others would extract the information from the source and return it to the digital library system, and other systems would summarize information from multiple sources. This system also used an economic system of fee-for-service charged by each component and across servers to handle load balancing problems for the queries made on the system.

With the explosion of the Internet as a source of information, the work that had once focused on digital libraries is applicable to a much broader set of information sources and applications. It is predicted that the number of Internet users will increase to over 100 million in less then 5 years [65], and the number of individual pages of content on the web is more than ten times that number. A large amount of the work done on digital libraries has now found applications in this much larger domain. As the quantity of data available has increased from the Internet, so has the variety of the information sources. Most of the current research in information extraction and integration has focused on two tasks: the creation of *wrappers* and information gathering agents.

A wrapper is a system that takes raw information directly from an information source on the internet and returns a set of data, which can be processed by an inference engine like entries in a database [2][3][4][6][49]. For example, a wrapper might take a list of currency values from a HTML encoded table and allow a system to calculate exchange rates from on type of currency to another.

Information gathering agents use planning and knowledge about the capabilities of a set of information sources on the Internet to break down complex queries into a set of simpler queries that can be sent to information sources on the Internet [20][21][28][29][53][56][73]. For example, an information gathering agent might take a home address and a type of cuisine and use this information to make a series of queries to a set of restaurant sites and map sites to find the closest restaurant that serves that specific type of cuisine.

Many of these information gathering systems are based on collections of specialized systems that interact with each other in real-time and that can be expanded to access new sites [11][22]. This architecture is very similar to the digital library systems developed at the University of Michigan. Value-driven information gathering also uses a similar architecture for converting raw data from the Internet into evidence that can be incorporated into the decision model. The contents of the extraction systems used by the value-driven information gathering system are considered black boxes that convert raw data from the Internet (a web page) into a set of records that can be incorporated into the decision model. These wrappers are a set of specialized systems that perform specific tasks on the raw data and pass the processed data to other wrappers or to the value-driven system for integration into the decision model.

One of the major contributions of value-driven information gathering is to the area of information integration. Value-driven information gathering defines an architecture for incorporate information into a decision model that allows a system to probabilistically reason about the state of information that has not yet been returned. Value-driven information gathering also provides a mechanism for incorporating the cost of gathering information into this model. This is important because up to this point much of the work in integrating information for various sources has relied of ad-hoc algorithms that are not provably correct. The most rigorous of these systems has relied on a simplified version of Horn-clauses. The advantage of using a horn-clause system is that the results produced by the system are provably correct, but the system is not able to make probabilistic decisions.

This dissertation does not add any new results to research in wrapper creation, except to offer a test bed that uses wrappers extensively. At this time there are at least four automatic and semi-automatic wrapper creation systems being developed [1][4][35][52]. The value-driven information gathering systems described in this dissertation has simply implemented the portions of these wrapper systems which we have needed to work in our test domains.

Another approach to integrating information from varying sources is attempting to develop common ontologies that can be used to map the information returned by various sources into a common framework. One of the advantages (and challenges) of using the internet is the large number of information sources which have information on the same object. If a system needs to learn a particular feature of a product, there are generally several dozen sites, which provide this information. Each of these sources uses its own terminology to convey the value of the feature that is important to the system. For example, if we were talking about rating a quality of a product, such as price, some sites might use the words *low, medium* and *high*, while another might rate the price of the product on a four star scale, or use *excellent, good,* and poor. These sites are all attempting to convey the same information, but are using different symbols to represent this same piece of raw data. This has lead to research in automating the construction of systems that can integrate this information [13][57][15] into one usable scale. Constructing these ontologies between different information sources is an extremely difficult task, which relies upon some knowledge of both the product and the domain, but by leveraging a common set of features that all of the information sources must share, these systems are able to construct ontologies, which can translate the description of features into one product based scale. This research is a good complement to value-driven information gathering because it provides a mechanism for integrating the raw information returned by information sources directly into the decision model using a pre-defined set of states.

Another approach for integrating information is to evaluate the structure of the documents themselves to reason about the relationships between them. The WHIRL system [14][48] attempts to convert web sites into database objects by modeling information sources as relations and using the similarity between the relationships to join different web sites.

Previous work in value-driven information gathering has also created techniques for reconciling data from different information sources based on decision model and evidence nodes [33]. The features described by different information sources can be modeled as evidence nodes in order to reconcile different scales, biases, and ranges in feature descriptions that one site might have from another. For exampled, evidence nodes can be used to integrate data from two information sources that use different scales: One source may have a rating scale from one to five while another source has a rating scale from 1 to 10. It is also possible to use evidence nodes to account for the bias that one information source may have that another does not. For example, a site may be particularly friendly to products from a specific manufacturer. Evidence nodes can be used in this case to have its rating lowered when compared to the same rating from another site. This is done by using the automatically constructed conditional probability table to translate the value from the information source into a universal scale. This universal value is used to calculate the influences that this evidence node has on the variable node for the feature in the decision model (see section 3.1.3 and 6.5 for more details). Evidence nodes also allow the system to represent the reliability, or trust, the system has in the information source.

2.4 Integrated systems

This section describes several integrated systems that are closely related to value-driven information gathering. All of these systems gather information from the Internet and return information to the user from multiple sources. While most of this work focuses on extraction of information, they also deal with the integration to some degree.

2.4.1 High-level information agents

Etzioni at the University of Washington has created several high-level agents that combine information from a number of sources on the Internet in order to give the user higher quality information [23][59]. Etzioni views the Internet as a growing ecology, which is currently inhabited only by *raw information* and *information herbivores* [24]. An example of a raw information source would be any page of information, it contains raw facts that the user must find and process them self. An example of an information herbivore would be a service like AltaVista. This type of agent processes all of the available information on the Internet in a slow methodical manner. It attempts to find all pages by following links and it builds a massive database of information that a user can use to find specific information they need.

Etzioni wants to build softbot agents, which he classifies as *information carnivores*. An information carnivore would use the services currently available on the Internet as tools to further process information before it is passed on to the user. He believes there are several benefits to building such systems:

1. **Systems can be client based** – Unlike an Internet search engine, which has to deal with millions of pages, an Etzioni softbot agent only has to deal with a few dozen web pages, which are returned by a search engine or other Internet agent. Because the system does not need to maintain a large database of web sites or constantly update that database as an ongoing task, the Softbot can reside on the client computer instead of a centralized server. For example, AltaVista could not be client-based since it needs to look at all of the web pages on the Internet to find matches, and the time required to do this is so great that it is worthwhile to maintain this information in a large database to cache the results. An agent like Etzioni's MetaCrawler [67] could be run by a client since it accesses only a few dozen sites (namely, other search engines), and then analyzes the top pages that each search engine returns. The database of information that the agent must maintain is small and can reside on an individual's computer.
2. **Systems can automate redundant work** - For example, Etzioni's ShopBot agent [18] could rapidly find and process a large number of sites to find the best price for a particular item. Compared to human agents, the ShopBot agent was faster and produced higher quality results. Etzioni credits this mostly to the fact that humans would quickly tire of the process of repeatedly searching for the price and give up after a few sites were found.
3. **Agent can be processor intensive** - Because the agents can reside on the client side, each agent can run more processor intensive tasks than a system that was based in a large centralized server. Some of these tasks include text parsing, database querying, or attempting multiple queries. For example, the ShopBot agent and the Ahoy agent both analyze the pages they receive before passing information on to the user, the ShopBot in order to find prices and make sure that

this is the correct product, and Ahoy to find the homepage of a specific user. Both of these tasks take the computer several minutes to accomplish for a small set of Internet sources. It would impossible for a centralized system to do this unless it received very few users or it was an extremely fast and expensive machine.

In some sense, value-driven information gathering is an extension of some of Etzioni's work. A value-driven information gathering system uses low and medium level Internet resources, is computationally intensive and can be client based. Value-driven information gathering systems differ from Etzioni's softbots in that they are resource conscious and attempt to prioritize potential information sources by their value to the overall decision. One key difference between value-driven information gathering systems and many of Etzioni's softbots is that the latter specializes in coverage: Finding every source that has the price of an item, or checking every location returned by a search engine in order to find a person's homepage on the Internet. Value-driven information gathering considers the available resources in order to query information sources until it is no longer advantageous to do so, using the resource cost of making a query and the decision model.

2.4.2 Information Extraction from Heterogeneous Sources

The Stanford Information group has constructed a system called Infomaster [21][28][29]. This system takes queries from a user and uses the Agent Communication Language in order to convert a request into a set of queries and operations on a number of independent databases. They call this system a virtual information system in that the system does not actually contain any information itself, but instead serves as an intermediary between the user and a number of on-line resources that have different interfaces and contents. For example, a user may want to find all of the houses for sale with a two mile radius from Stanford. The Infomaster system queries two on-line databases, the first is a listing of housing available near Stanford, and the second is a street map database that gives distances between any two points in the Stanford area. By using both of these simple databases, the Infomaster system is able to answer a number of sophisticated queries. The advantages being that the user does not have to use two different interfaces, and the user does not have to query the street map database for each available house.

The Agent Communication Language used by Infomaster is a simplified predicate language, used to translate complicated queries into a series of simplified querying actions. For example, a rule to find the phone number of an advisor might be written as follows:

phone(x, z) :- office(x, w), office(y, w), phone(y, z)

In other words, if we can find the phone number of someone in the same office, then that is the advisors phone number as well. Commands can also be broken down into information about possible sources, for example:

phone(x, y) :- whois-phone(x, y) or dorm-phone-db(x, y)

These rules in turn are broken down into actual database queries and extraction methods. By defining a predicate language for navigating and combining database information, the Infomaster system can plan information gathering using standard logical inference techniques.

Infomaster also differs from value-driven systems in that it does not take resources or information source costs into account. The predicate logic system does offer some sense of prioritizing in the case where alternate rule expansions are offered, but it does not allow the system to return meaningful partial results if the system does not have the resources to fully answer the query.

2.4.3 Design-to-criteria information gathering

The multi-agent system lab at the University of Massachusetts has been working on an advanced information gathering framework called BIG, which uses TAEMS for describing information gathering plans [73] [56]. TAEMS is a representation of flexible tasks that has been used in several applications. The BIG system uses TAEMS task

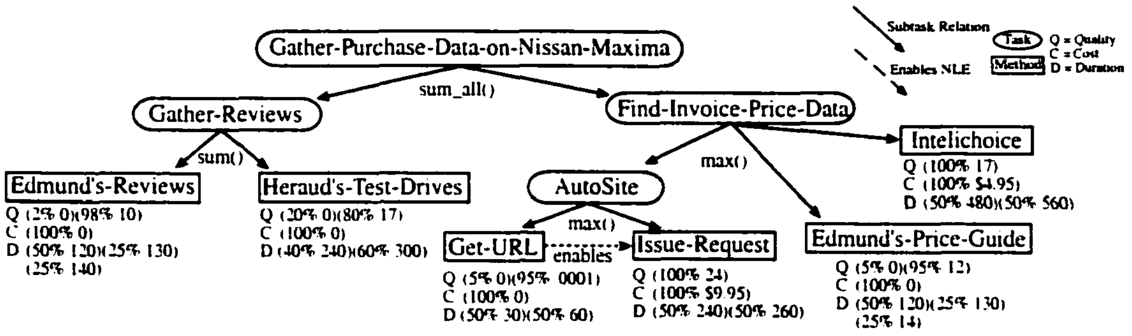


Figure 2.8 - An example TAEMS task structure for Gathering Auto purchase information

structures (see **Figure 2.8**) in order to plan near-optimal information gathering plans, which take into account cost, duration, and quality of results. This computation is expensive, but the multi-agent group has been able to use a satisficing methodology to meet the goal of the task structure in low-order polynomial time without much loss in quality.

The group has developed several information gathering applications to support decision making [55]. These systems use the TAEMS task structure to represent and plan the information gathering process. They also use the extracted information to monitor and change the plan. Their information gathering agent has a number of distinct components:

RESUN Planner

The RESUN planner coordinates the information gathering process. It receives the information gathering goal from a user or high-level system and builds alternative ways to reach the goal. It takes into account cost, duration, and quality. It also takes into account the uncertainty of the information gathering plans.

Task assessor

The task assessor builds TAEMS task structures from the RESUN planner for the Design-to-Criteria scheduler to evaluate.

Design-to-Criteria Scheduler

The Design-to-Criteria scheduler builds a task schedule used by the Information Gathering Planner to collect information used for the decision. It takes the TAEMS task structure and converts it into a plan of Internet queries, which the information gathering agent can execute.

Internet retrieval interface

A low level interface between the Information Gathering Agent and the Internet.

Text processing tools

These tools are used extracting database-like data from a text document.

The BIG information gathering system has more in common with value-driven information gathering than any of the other systems described in this section. Both systems reason about resources, costs, and quality. The TAEMS task structure allows the BIG system to build a set of alternative information gathering plans, which are evaluated to determine a satisficing plan to execute. Value-driven information gathering systems, on the other hand, use the decision model and the response expectation of an information source to calculate a numerical score for each potential information source at each step in the information gathering process. Compared to the approach used by the BIG system, value-driven information gathering uses a myopic approach to deciding which information source to query next at any given time. BIG also combines quality or value of information using simple functions (quality combination functions) that cannot even approximate the correct value of information. Our reason for using this approach is simply one of complexity. Value-driven information gathering deals with more potential information sources than the BIG system. The large number of potential information sources, as well as the variance in the data and time at which that data may return makes generating a querying plan at the outset of the gathering session unfeasible. **Table 1.1** shows how variance in the data returned by an information source can affect the remaining queries made during the information gathering session. The combinatorial space representing all of the possible evidence that could be returned as well as all of the possible times at which that information could be returned is too large to explore in the process of creating a general-case information gathering plan for the value-driven system to execute.

2.4.4 Cooperative information gathering and processing

Cooperative information gathering and processing is defined as the process of constructing a set of specialized agents, which are able to work together to answer complex queries. Onn Shehory has developed this architecture as well as several systems that work on real-world problems [69]. For example, in a cooperative information gathering system, the designer would build several agents in order to answer a specific class of complex queries. These might include several agents for accessing a set of sites, agents for combing specific results, an agent for coordinating the task and an agent for interacting with a user. The advantage to designing an information gathering system in this way is that agents are small, easy to write and debug, and the system is easily extensible.

The disadvantage to such systems is that forming coalitions of agents is an exponentially large search space of potential agent groupings. Shehory has dealt with this problem by developing a distributed method of building coalitions and a set of specialized agents that manage the matchmaking process between groups of agents. When an agent joins the system, it advertises its capabilities to a matchmaker agent, which in turn connects a requesting agent to the proper serving agent (or set of serving agents). When it is time to form a coalition of agents, each agent divides the grouping problem by solving the case where it is a member. In this way, adding agents to the system does not increase the computational load on each agent exponentially.

Shehory's coalition system also does not take querying cost or resources into account, but a value-driven information gathering system could be designed using the agent system that Shehory describes in his work. Unfortunately Shehory's framework is very open, so even though a value-driven information gathering system could be built in his framework, it would not make the system more flexible, except to add distribution of the computational resources required for decision modeling and managing the information sources database. If these computations are too expensive, they could be solved by adding parallelism directly instead of developing an entire agent system.

2.4.5 The Ariadne project

The Ariadne project [2][3][4][6][49] is a system that automatically adds layers to Internet resources in order for them to be accessible as databases. The Ariadne system does this by automatically building wrappers to access the semi-structured information of web pages and by building a querying plan to answer complex queries. Knoblock's experimental application is an interface layer on top of the CIA factbook. The CIA factbook contains pages on hundreds of countries, but it has no built-in querying engine. The Ariadne project has built a querying layer on top of the factbook by building wrappers for specific items and a planning engine for querying the correct pages to answer a query.

The Ariadne project does not use a response expectation to represent how quickly an information source will return. Instead, it tries to minimize the total number of queries made in order to minimize the total time it takes to answer a complex query. It is not necessary for the Ariadne system to reason about the response time of different queries because the Ariadne system queries resources from one server, so all the queries should have similar response times.

The total number of queries sent to the server is reduced by building and analyzing a *discrimination matrix* that is used to determine which information sources are available at run time and what information they contain. The system reasons about the domain using a variant of first-order logic that allows the system to represent unary and binary relations between objects. The discrimination matrix is then used to build and alter plans so that it minimizes the total number of queries that need to be made. For example, using the CIA factbook we might want the GNP of all NATO countries. The Ariadne system first uses the discrimination matrix to eliminate the non-NATO countries before requesting the GNP. This can drastically reduce the total number of queries needing to be made.

The Ariadne project differs from value-driven work in that it returns complete results and it does not have a very sophisticated response expectation system. The Ariadne work expands on the Stanford work by doing some reasoning about resource usage and trying to reduce the total number of queries made by looking at the structure of the query. Ariadne systems could provide good information sources for value-driven information gathering systems because they make web resources act more like databases that can be sent complex queries and reason about the time it will take to respond.

2.5 Recursive information gathering plans

Recursive information gathering plans [20] use Horn rules to answer a query using several information sources. The user creates a query using a set of *virtual* relation names that represent the query that the user want answered. These virtual relations are then mapped to a set of source relations using sets of conjunctive queries. The example used in the paper is to create a set of virtual relations that connect papers and authors.

$$db_1(P,A):-paper(P),author(P,A),ai(A) \tag{2.6}$$

Paper, *author*, and *ai* are the virtual relations that can be used to make queries and db_1 is the source relation that actually exists as an external information source. This representation is very similar to those used in the Infomaster system [21] and others [53]. After a query has been made using the virtual functions, the system will recursively generate new virtual and source relations until only source relations are left. At this point, the system can query the information sources and return the results of the query. The system takes advantage of functional dependencies to simplify the task of answering queries that otherwise would have no solution. For example, given the following set of virtual relations:

$$v1(P,C,Y):-conference(P,C),year(P,Y) \tag{2.7}$$

$$v2(P,L) : -conference(P,C), year(P,Y), location(C,Y,L)$$

The system can be given the query shown in equation (2.8) and return a valid response by finding *any* paper at IJCAI in 1991 and finding the papers' location.

$$q(L) : -location(ijcai, 1991, L) \qquad (2.8)$$

$$answer(L) : -v1(P, ijcai, 1991), v2(P,L)$$

This is possible because every paper is presented at one conference in one year. Using functional dependencies allow this system to answer queries that a system that only expanded Horn-clauses could not.

The paper also presents a method for representing specific binding patterns for source relation. For example, equation (2.9) describes a source relation that can only give the rule set Y if X is bound to a value.

$$v_2^{bf}(X,Y) : -cites(X,Y) \qquad (2.9)$$

The system can construct querying plans in which another source relation returns a set of items to bind to v_2^{bf} and the results are used by other clauses. This allows the recursive system to build plans that previous Horn-clause systems could not.

Recursive information gathering plans allow for the uses of function dependencies and limitations on binding patterns that are often present in the Internet domain. The next step for the system is to add ordering predicates so that the Horn clauses can be sorted in a way to reduce the number of queries that need to be made.

Recursive querying plans are a good compliment to a value-driven information gathering approach. Like many of the systems described above, the focus is on complete coverage of the available information sources and the costs of querying an information source relation is not considered. The advantage to this technique is that it expands on the capabilities of a Horn-clause approach, which is founded in predicate logic and thus has a large body of research to draw on. However, it expands those capabilities to find a better match to the environment of the Internet. I suspect that these approaches to information gathering will become a focus of many researchers in the future and will become the major competing approach to a value-driven information gathering approach. The main assumption made by these techniques is that a system will have the resources to do an exhaustive search in order to find the optimal result or the complete set of results. The value-driven approach assumes that it will not be feasible to exhaustively query the set of potential information sources. This leads to a more probabilistic approach to both the result (using a decision model instead of predicate logic) and the querying technique (determining the cost of querying an information source).

2.6 Discussion

The main concept behind value-driven information gathering is the determination of the value of a query. The value of a query is in turn defined as the expected gain in decision quality that comes from the evidence returned by an information source and the expected cost in resources from making the query. Given an accurate estimate of the gain in utility from querying an information source, a value-driven information gathering system can effectively use the resources and information sources it has been given to make a decision. Calculating the value of a query will be discussed in detail in Chapter 4 but the two main factors that influence the value of a query are a notion of the value of the information that the query will return and the resource cost of making the query.

Decision theory provides a method for determining the improvement in decision quality from acquiring a piece of information. This is the first part of the value of a query calculation. Resource-bounded reasoning provides a method for calculating the cost of actions that take time to return a result and their effect on the output quality of the overall system. Using both of these techniques, the value-driven system can make a formal analysis of the real benefit from making a query to an information source. Many of the problems that other information gathering systems need to deal with are already solved by combining these two techniques. For instance, the integration of multiple pieces of evidence has already been solved in decision theory, and selecting information queries that best match the resources given to the system is addressed in resource-bounded reasoning. Merging decision theory and resource-bounded reasoning creates a comprehensive approach to information gathering driven by the value of a query.

The five integrated systems presented in section 2.4 all deal with extraction of information from multiple sources, integrating that data, and returning the processed information to the user. All of these systems, with the exception of the design-to-criteria system, do not consider resources when querying and returning information. Most of these systems are also focused on browsing and collecting information instead of acting on that information. Without a mechanism to gauge the value of the information being gathered or the cost of gathering it, it is very hard to develop gathering strategies that go beyond maximizing the amount of information collected.

While there may be cases where this solution is the best approach, as more information becomes available to machines using the Internet as a source of information, the ability to screen and analyze the benefit of gathering information will become increasingly important. Value-driven information gathering offers an approach for reducing the information gathering process by determining the effect of querying an information source by the effect on the quality of the decision and the resources used.

Using decision theory and resource-bounded reasoning as methods for calculating the value of a query, information integration and adaptability to various resource constraints may be included in determining the output quality of the overall system. Another benefit of using an influence diagram to represent the decision being made is that the system automatically generates a gathering plan based on the decision. It does not need to be given an implicit plan for collecting information in order to make a decision.

What these other systems focus on is the amount of information being gathered as opposed to its quality and cost. Value-driven information gathering attempts to solve the problem of directed information gathering with a more complete model of the gathering process. The decision model and the cost function drive the creation of the gathering plan.

CHAPTER 3

PROBLEM STATEMENT

The central problem for value-driven information gathering is determining which information sources to query in order to maximize the quality of a decision given a set of time and monetary resources. This chapter will define this problem more formally by giving an overview of the value-driven process, from the creation of a decision model to the return of a decision by the system. Where applicable, definitions will use the

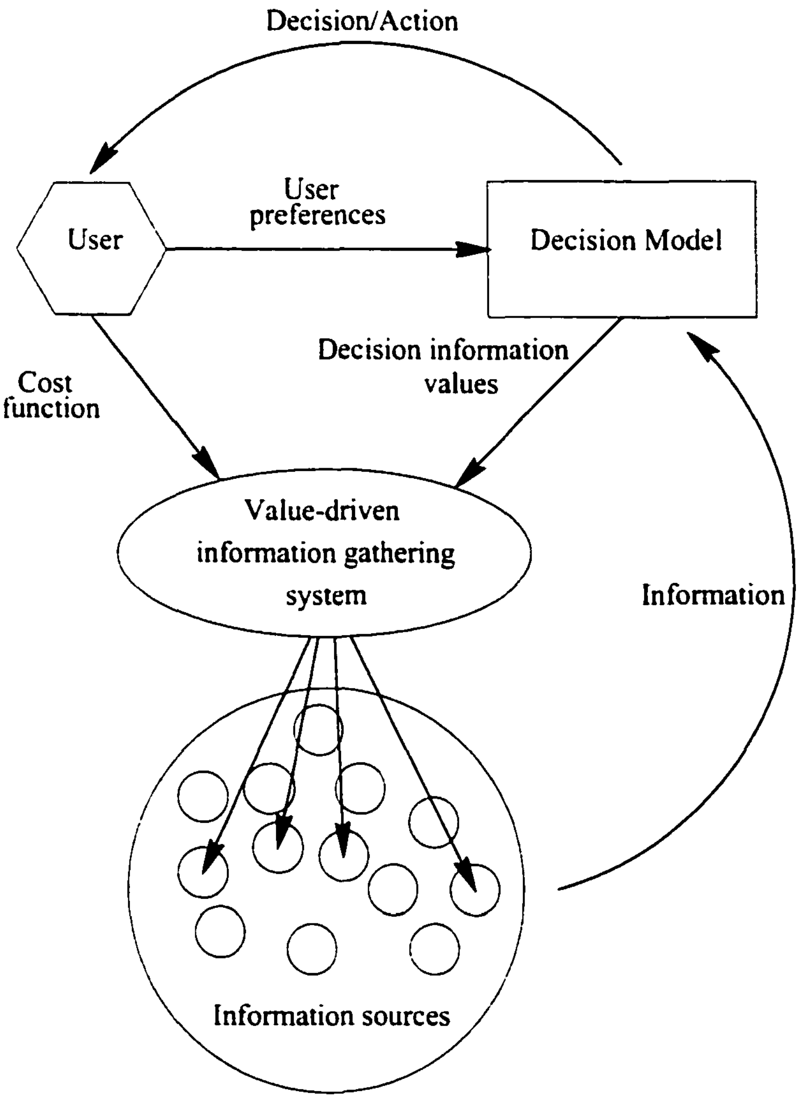


Figure 3.1 - An overview of the value-driven information gathering process

mathematical terms used in Chapter 4, where the value-driven information gathering algorithm is described in detail.

Figure 3.1 shows an overview of the value-driven information gathering process. The value-driven information gathering problem is how to maximize the decision quality given a decision model, a set of information sources, and a cost function. We have focused on a specific class of decision model in which the decision system must choose one instance of an object from a set of instances (including choosing none). This class of decision model is used in product selection. The process of developing and executing a value-driven system can be broken into four steps:

The first step, which is done off-line by an expert in the domain, is the creation of a decision model. The decision model contains a set of connected probabilistic nodes and a utility function that can evaluate the utility of selecting one object from the set of choices.

The second step, which is done at the beginning of the gathering session, is for the user of the system to specify a set of individual preferences and assign the resources that the value-driven system may use to make a decision (the *utility function constants* and the *cost function*).

Step three is for the value-driven system to begin the gathering process. This involves calculating the value of a query for each potential information source and querying the information source with the highest value. Step three is repeated until the cost of continuing the gathering process outweighs the improvement in decision quality that could be made by continuing to gather information.

The fourth step is for the value-driven system to return a decision to the user. The decision is based on the estimated utility of selecting each instance available to the system and comparing it to a baseline choice (generally, this is doing no action). The instance with the highest estimated utility is recommended.

The rest of this chapter will describe the terms used in formulating the value-driven information gathering problem. Section 3.1 describes the decision model used by the value-driven system, section 3.2 describes the cost function, section 0 describes the information sources used to gathering information and section 3.4 describes the gathering session and the summarizes the value-driven problem.

3.1 Decision Model

The focus in this work is on a particular class of decisions in which a system must select one object that best matches the user's objectives. Each object is an *instance* of the decision model. This instance in turn is composed of two parts, a belief network representing the beliefs the system has about the object, and a utility function that takes this set of beliefs and converts it into an expected utility for selecting the object. For example, in a product purchasing decision, each instance would be a brand of product. The decision model used in value-driven information gathering could be transformed into a general influence diagram, but the structure of the decision model increases the speed of the computation and reduces the complexity for the designer. **Figure 3.2** shows the general structure of the decision model for a digital camera. **Figure 3.4** shows a specific instance of a restaurant in the restaurant decision model. A digital camera purchasing decision and the restaurant selection will be used as examples throughout the chapter.

In **Figure 3.2** each gray rounded rectangle on the left represents an instance of a particular brand of digital camera from which the system will choose one (or none if the expected utility for each camera is below zero). Each instance has two parts: a belief network that represents the set of features for the object (the rectangle) and a utility function that takes the values of features and returns a utility for selecting that instance. **Figure 3.3** shows a detail of the belief network for each digital camera instance. The utility score for each digital camera instance is based on several factors: the cost of the camera, the storage capacity, whether it has a flash, the size of the image taken by the camera and the compression quality of the images.

The rectangle on the right side of the figure (*select one or none*) represents the purchasing decision selected by the decision model. The decision model may choose one of the instances to recommend for purchase or select none if the expected utility for all of the cameras is below zero utils (utility is measured on a scale of *utils*). If there are

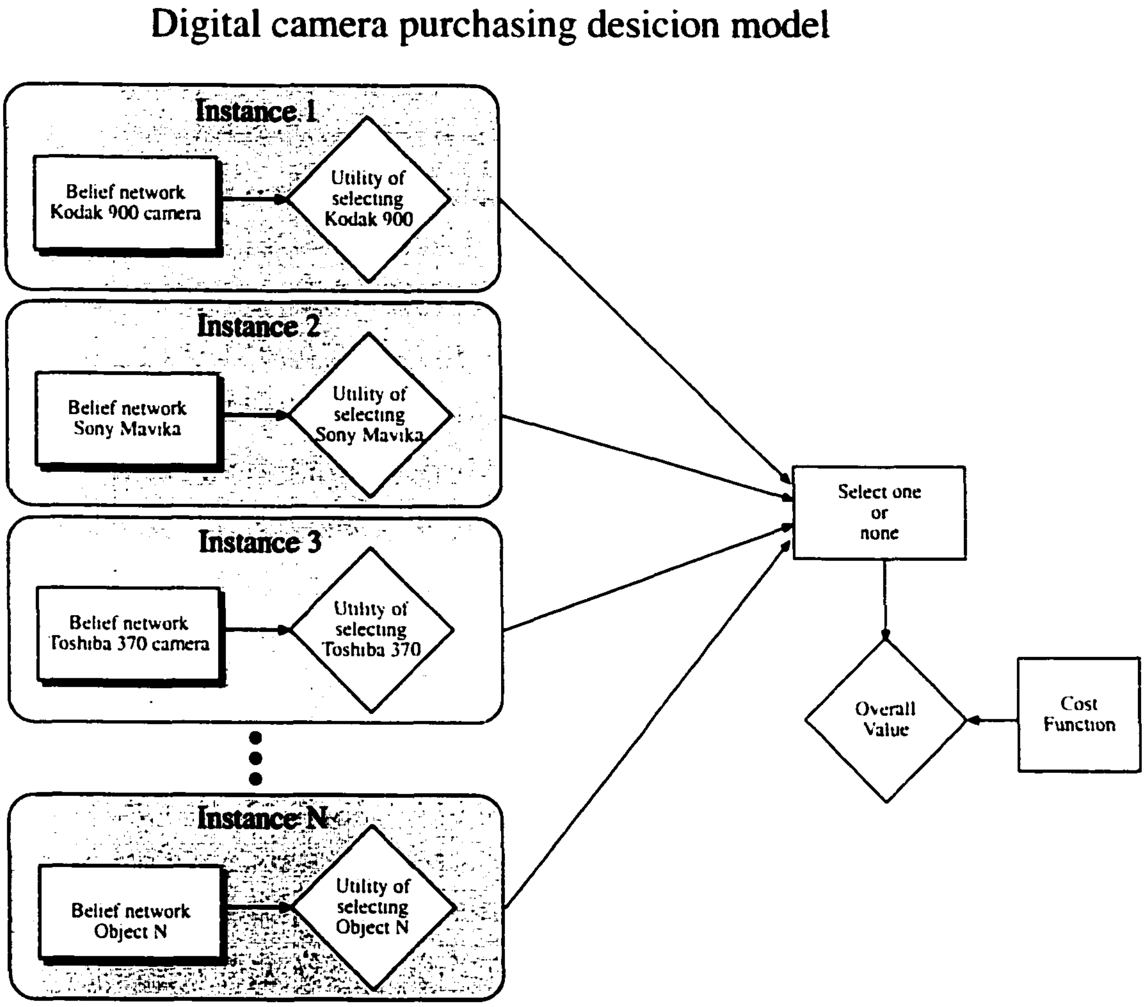

Digital camera purchasing desicion model
Instance 1
Belief network Kodak 900 camera
Utility of selecting Kodak 900
Instance 2
Belief network Sony Mavika
Utility of selecting Sony Mavika
Instance 3
Belief network Toshiba 370 camera
Utility of selecting Toshiba 370
Instance N
Belief network Object N
Utility of selecting Object N
Select one or none
Overall Value
Cost Function

Figure 3.2 – Digital camera decision model

multiple instances with positive expected utility values, then the instance with the highest utility score is selected. The *cost function* calculates a cost in *utils* for the resources used thus far in the gathering session (time, money, computational resources…). The *overall value* (the diamond in the figure) is equal to the expected utility of the selection decision minus the utility cost of the resources used in making the decision during the gathering session.

To the right of the overall value is the cost function. The cost function defines the cost to the system of spending resources gathering information. The cost is based on the time spent gathering resources and the monetary cost of querying information sources.

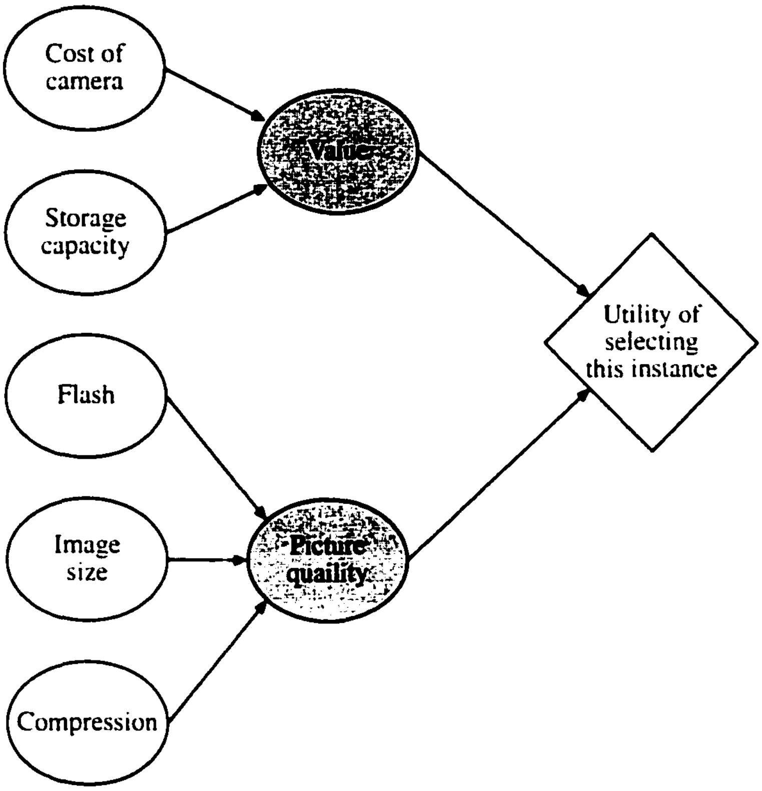
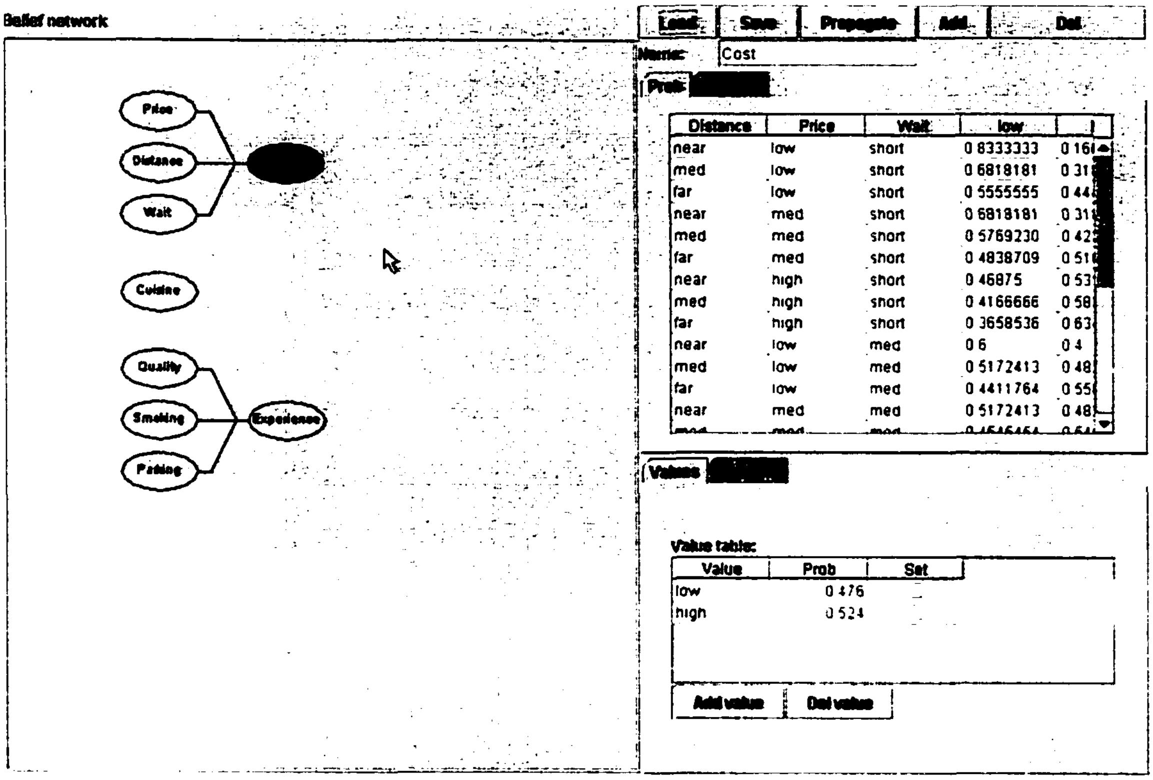


Figure 3.3 - Digital camera belief network for one instance

Figure 3.4 shows a specific instance of a restaurant in the restaurant decision model. The window on the left shows the belief network of variable nodes for features associated with evaluating a restaurant. For example, a restaurant has a variable node representing the average price of purchasing a meal at the restaurant. This variable node can have three possible states: Low, Medium, or High. Just because there are only three states which the variable node price can have, do not assume that we are specifying the granularity of the variable. This node will never actually be instantiated, so it will always have a probability distribution, which can represent any value in the range from low to high.

In this belief network, there are also two high-level variable nodes: *Cost* and *Experience*. These high-level variable nodes were created to represent an estimation of the negative and positive aspects of dining. The *Cost* node, which represents the negative aspects, is influenced by the *price*, the *distance* to the restaurant, and the *wait* to be seated. The *Experience* node, which represents the positive aspects of dining, is influenced by the *quality* of the food, if the restaurant is non-*smoking*, and if there is



Name: Cost

Distance	Price	Wait	low	
near	low	short	0 8333333	0 16
med	low	short	0 6818181	0 31
far	low	short	0 5555555	0 44
near	med	short	0 6818181	0 31
med	med	short	0 5769230	0 42
far	med	short	0 4838709	0 51
near	high	short	0 46875	0 53
med	high	short	0 4166666	0 58
far	high	short	0 3658536	0 63
near	low	med	0 6	0 4
med	low	med	0 5172413	0 48
far	low	med	0 4411764	0 55
near	med	med	0 5172413	0 48

Value table:

Value	Prob	Set
low	0 476	
high	0 524	

Figure 3.4 - Restaurant decision model

parking. These high-level nodes do not have any inherent reason for existing in this belief network, other than the expert who created the belief network placed them in the network.

The table on the top right of **Figure 3.4** is the conditional probability table for the *cost* node. You can see that the values of the parent nodes (*price, distance,* and *wait)* influence the probability distribution of the value of cost from *low* to *high*. Again, it is important to point out that since the variable node *cost* will never be directly instantiated, the number of values that the node can take has very little meaning. The probability distribution of the node allows the node to contain a real-value, which varies between zero and one.

Finally, the table on the bottom right of **Figure 3.4** is the probability distribution for the node *cost*. At this point, with no evidence for the restaurant (this screen shot was taken before the gathering session had begun) the probability for *cost* being low is 47.6% and 52.4% for *cost* being high.

3.1.1 Prior probability distributions

In any system that uses a belief network for making decisions or modeling the world, a moment must be spent to discus prior probability distributions (priors) and how they affect the system. Priors are the unconditional probability distributions for leaf nodes that have no parents. These probability values cannot be conditioned on any other node in the belief network when it is constructed. Whereas the conditional probability tables for the internal variable nodes in a belief network can often have some basis in the mathematical relationship between the variables, the conditional probability table for the leaf nodes can often only be based on statistical experience in the domain. One of the assumptions made for value-driven information gathering is that the belief network used in the decision model will be created by an expert in that domain and that the expert will have some knowledge of what the typical probability distribution will be for a leaf node. Fortunately, if this is not the case, the system will not fail, but the performance of the system will be below that of a system with a more accurate set of prior probability distributions. The decrease in performance will come from two factors: First, the decision model itself will make incorrect decisions for leaf nodes that do not have evidence attached to them. Second, the value-driven information gathering system will make less optimal decisions about which information sources to query based on the value of information for that node.

For example, if the belief network contains a leaf node with a prior probability distribution of True = 0.999 and False = 0.001, the value of information for learning the state of the node is extremely low. The value of information for this node is low because it is extremely likely that you will learn that the state of the node is True, which will have very little impact on the probability distribution of the other nodes since the probability distribution will only be changing from 0.999 to 1.000. If the probability distribution for the node is actually True = 0.1 and False = 0.9, than the system will very rarely discover the true state of the node. Instead, it will choose the incorrect value without gathering evidence.

The primary solution for this is for the designer of the belief network to be cautious in how uneven they make the prior probability distribution for the node. If the prior probability distribution for the node is an even distribution, then the value-driven system will have a greater chance of gathering evidence that influences the probability distribution of the node and propagates the more accurate value through the network. Although this does not reduce the problems arising from the network making incorrect assumption when there is no evidence, it will increase the likelihood that the value-driven system will query an information source that will return evidence that influences the probability distribution for the node. It should still be stressed that our assumption is that the creator of the belief network used by a value-driven information gathering system will be accurate. Finally, it should also be noted that there are several techniques for constructing prior probability distributions as the system interacts with the real world.

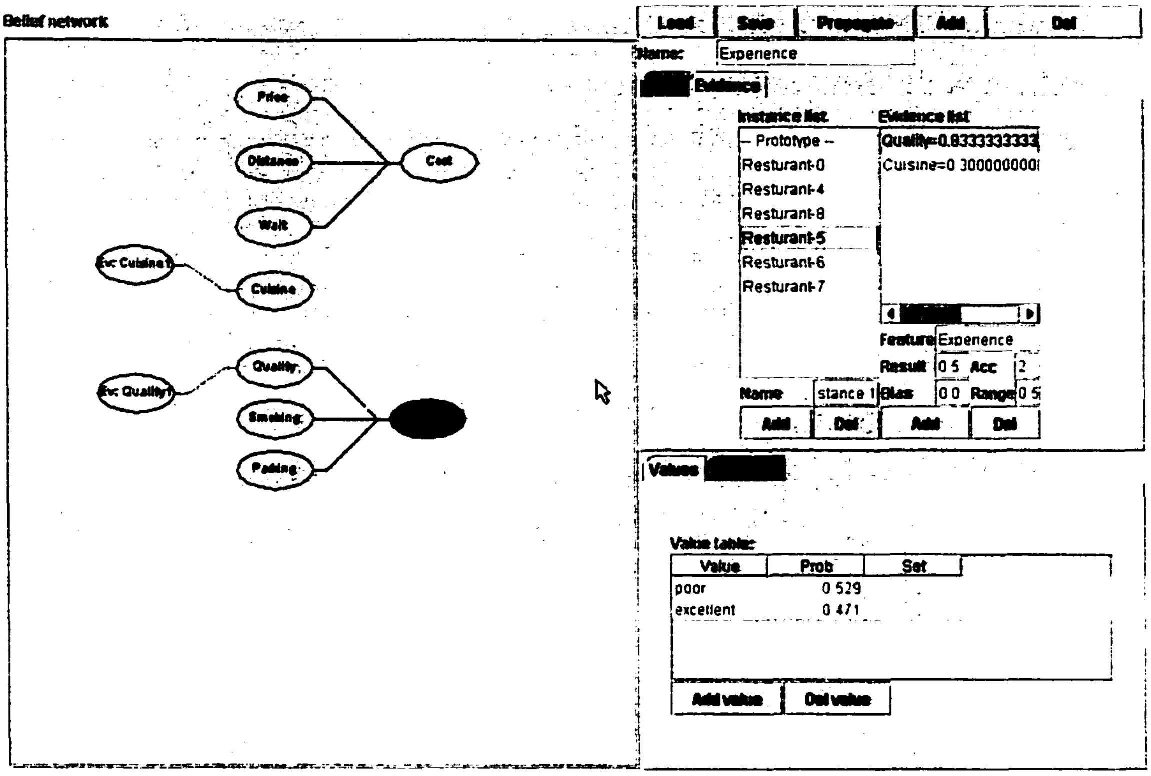
3.1.2 Instances

Each individual object that the value-driven information gathering system can select from is called an *instance*. Each instance has three components: A belief network used to propagate and reason about the object, a set of evidence that has been retrieved that applies to the object, and a utility function, which takes the probability distributions of variable nodes in the belief network and returns the expected utility for selected the instance.

Separating each instance into an independent belief network allows the value-driven information gathering system to reduce the computational cost of maintaining the belief networks for all of the instances. Changes to the probability distribution of a variable node in a specific instance cannot alter the probability distribution of the variable nodes in other instances.

The expert who constructs the decision model creates one *base belief network* and utility function, which is duplicated for each instance available[2] to the system. New

[2] Although none of the systems presented in this dissertation uses this capability, it is not necessary for instances to have the same belief network or utility function. The only



Name: Expenence

Instance list:
-- Prototype --
Resturant-0
Resturant-4
Resturant-8
Resturant-5
Resturant-6
Resturant-7

Evidence list:
Quality=0.8333333333
Cuisine=0 3000000000

Feature: Expenence
Result: 0 5 Acc: 2
Name: stance 1 Bias: 0 0 Range: 0 5

Value	Prob	Set
poor	0 529	
excellent	0 471	

Figure 3.5 - Restaurant instances and evidence

instances can be added to the decision model dynamically, as in the restaurant selection system (see section 6.6). Each instance can return the expected utility for selecting that instance at any time after it is created.

The process of replicating the belief network for each instance is not difficult. A database is used to maintain the conditional probability tables for each node in the belief network, the actual probability distribution for the nodes in each instance, and the evidence that applies to each instance. When a new instance is created (see section 4.3, 6.5 and 6.6) a new set of entries in the database are created with the initial probability distribution for an instance with no evidence. This task takes the system extremely little time to execute. When evidence for an instance is returned by an information source, the probability distributions and old evidence for the instance, along with the new

---

requirement is that each instance gives the expected value of selecting the instance from the set of instances.

evidence, are loaded into the belief network engine and the new probability distributions are calculated. A similar process is done when the value of a query is calculated for a potential query to an information source. The process of changing the particular instance used by the belief network engine is also an extremely quick process.

Figure 3.6 shows the belief network for an instance of the digital camera purchasing decision. Each of the oval nodes in the graph are variable nodes with a probability distribution and a conditional probability table that relates the probability distribution of this node to the probability distribution of the connected nodes (see section 2.1 for more detail on belief networks). The gray ovals are variable nodes used by the utility function (represented by the diamond in **Figure 3.6**) to calculate the expected utility of purchasing the digital camera. This belief network is replicated for each brand of digital camera that is being considered by the value-driven system. **Figure 3.5** shows the restaurant example with instances and evidence.

The belief networks used by each instance of the decision model are composed of variable nodes as described in section 2.1. In the belief network shown in **Figure 3.6** we have five leaf nodes (*cost of camera*, *storage capacity*, *flash*, *image size* and *compression*) and two internal nodes (*value* and *picture quality*). We had to set the prior probability distributions for the leaf nodes and this was done by randomly sampling a set of digital cameras available at the time. In **Figure 3.6**, the node *Cost of camera* can have a state of **$100-150**, **$150-200**, **$200-250**, **Over $250** with a prior probability distribution of (0.2, 0.3, 0.4, and 0.1, respectively). The importance of prior probability distributions in the execution of value-driven information gathering is described in detail in 3.1.1; the priors for the *Cost of camera* variable node were derived by looking at a ten randomly selected digital cameras available at the time. The node *Picture quality* can have the value of **low**, **medium,** and **high** based on the state of the nodes *Flash*, *Image size*, and *Compression*.

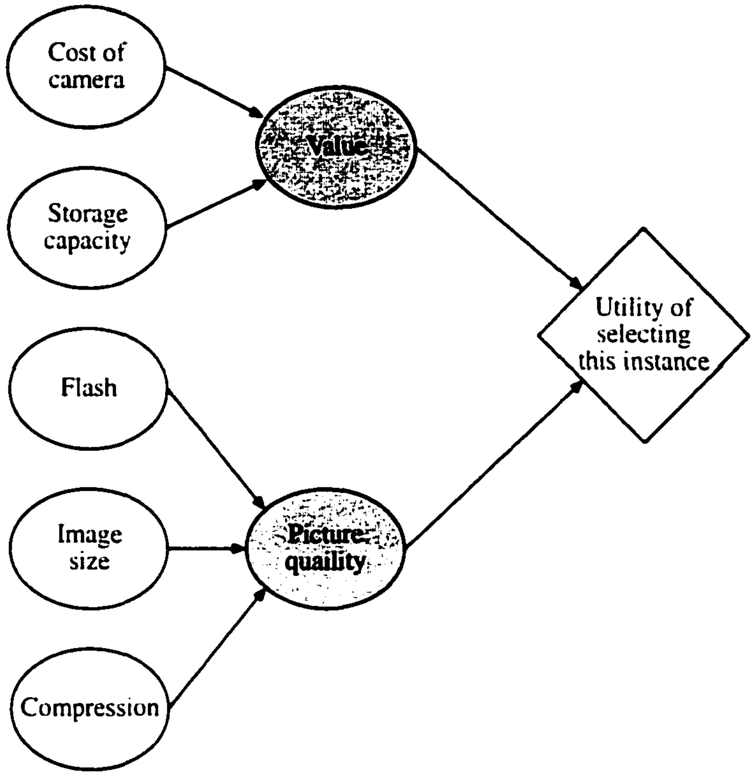


Figure 3.6 - Digital camera belief network

The probability distribution of each variable node in a belief network can be determined using the rules of probability and the evidence that has been returned by information sources. There are several well-known algorithms for determining the probability distribution for a belief network [12][58][62]. I used a message-passing version of the algorithm described in [62][3]. This algorithm was designed for use in poly-tree belief network topologies, which included all of the networks used in this research. This algorithm, when done as a message passing process can propagate the probability distributions of a belief network in linear time based on the number of nodes. The built-in memory management and garbage collection of a language like Java made writing the belief-network propagation algorithms much easier. This is because

[3] There was a flaw in the original printing of the book, but Russell distributed a corrected algorithm on the Internet, which I used in this research.

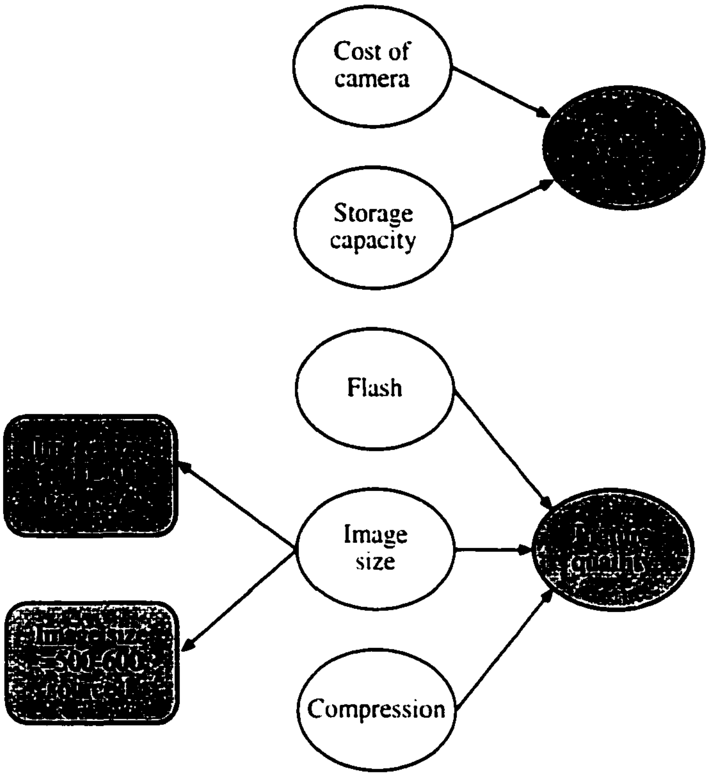


Figure 3.7 - Evidence collected for the image size

the algorithm creates and disposes of a large number of intermediary message objects that are used to calculate the new probability distribution.

3.1.3 Evidence Nodes

Evidence nodes are a restricted form of variable nodes that allows the value-driven system to deal with two important issues: The varying reliability associated with particular information sources; and combining conflicting information from multiple information sources. Evidence nodes have three restrictions, which simplifies their creation and propagation. First, evidence nodes only connect to one variable node in the belief network of one instance. Second, evidence nodes are assumed to be causally independent of each other. Third, evidence nodes have a conditional probability table that is calculated from three values: The accuracy, bias, and range of the information

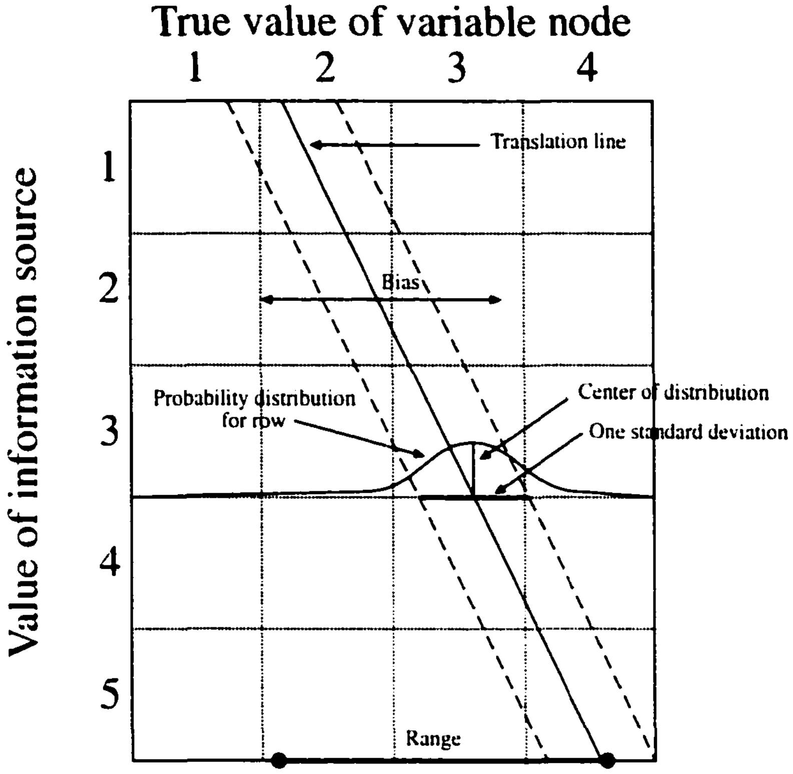

True value of variable node
1
2
3
4
Value of information source
1
2
3
4
5
Translation line
Bias
Probability distribution for row
Center of distribution
One standard deviation
Range

Figure 3.8 - Generating a probability table from an accuracy and bias

source. What these three terms mean and how they influence the conditional probability will be described in detail below.

In our digital camera example, as we collect evidence for a particular digital camera, new evidence nodes are added to the instances. The evidence nodes change the probability distribution of the features and thus the utility of selecting the instance. **Figure 3.7** shows a digital camera instance after two pieces of evidence about the image size have been returned by information sources 1 and 4 (the rounded rectangles). **Figure 3.5** shows evidence nodes added to an instance of the decision model for the restaurant decision.

The three values used to create the conditional probability table for an evidence node define a line relating the values that the information source may return to the states that the feature may have. The three values also define the "noise" associated with the evidence. **Figure 3.8** shows how the three values for the evidence node define the conditional probability table.

Accuracy

How accurate is the data being returned? We may trust information from one information source more then we trust information from another. For example, we may trust the weather prediction from the newspaper more then from our next door neighbor. The accuracy determines the error associated with the probability distribution for the row in the conditional probability table.

Bias

Is there a constant shift in the result reported by the information source? For example, a specific magazine may consistently rate all movies much better then they actually are. Bias allows the system to account for constant shifts as opposed to inaccurate results. The bias is equivalent to the offset of the line in **Figure 3.8**.

Range

What is the true range of the evidence being returned by the information source? Often times information sources only use a small portion of the values they may return. To continue the previous example, if a magazine can rate movies from one to four stars, but in actuality only rates movies as 2, 3 or 4 stars, then the range of the evidence returned by the magazine is only three although there are four possible rankings that it could give. The range is used to determine the slope of the line in **Figure 3.8**.

These three terms are collectively called the *reliability* of an evidence node. The reliability determines how the evidence node will influence the rest of the belief network when it is instantiated. **Figure 3.9** shows three examples of the conditional probability table for an evidence node.

Figure 3.9 (a) shows the conditional probability table for an evidence node with a high level of accuracy, a full range, and no bias. This evidence node represents a very reliable source of information. Notice that in the regions where the value of the

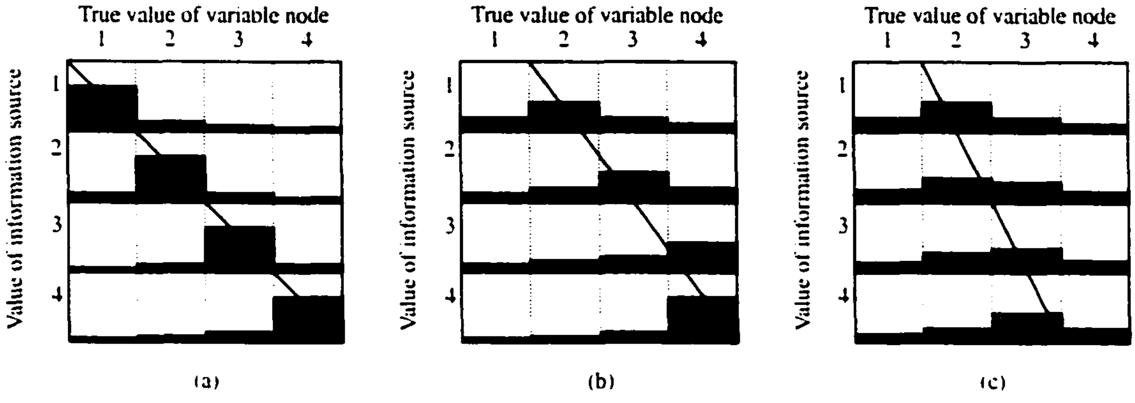


Figure 3.9 - Conditional probability tables for different types of evidence nodes

information source is equal to the value of the variable node, the bars has a very large value which quickly drops off as we deviate to the left or right. When we instantiate the value of the evidence node to a particular value, it will have a strong influence of the probability distribution of the variable node which connects to it.

Figure 3.9 (b) shows the conditional probability table for an evidence node with a negative bias and medium accuracy. The difference across the rows for the conditional probability table is not as large as they were for **Figure 3.9** (a). The conditional probability table also shows that the even when the value of the information source is 1, the most likely state for the variable node is 2. This evidence node has a negative bias compared to the evidence node in **Figure 3.9** (a). It consistently reports a lower value then the true value of the variable node. This evidence node is also less accurate in its prediction of the value of the variable node. The construction of the conditional probability table can compensate to some degree for the bias, but the lower accuracy can only be modeled by decreasing the influence that this evidence node will have on the probability distribution of the variable node once it is instantiated.

Figure 3.9 (c) shows the conditional probability table for an evidence node with reduced range and a low accuracy. Compared to the other evidence nodes in this figure, this node is not a very reliable indicator of the true value of the variable node. The more even the distribution of bars across a row is, the less influence the evidence node will have on the probability distribution of the variable node once it is instantiated. An instantiated evidence node with a completely flat distribution across each row would not alter the distribution of the variable node that connected to it. This evidence node also has a reduced range compared to evidence node (a) and (b). As the state of the evidence node varies from 1 to 4, the variable node state with highest probability distribution only varies from 2 to 3. This represents a source of evidence which over states the value of the variable node. For example, it would review medium valued products as either excellent or poor. The conditional probability table compensates for this be reducing the range that the evidence node influences.

The reliability for an evidence node may be determined using a line-fitting algorithm and previous examples from the information source, or the reliability associated with a piece of evidence may vary based on other factors. For example, evidence may be time dependent, the value of a stock becomes increasingly inaccurate the older the data. In addition, we can adjust for larger scales of time by applying bias to the information. For example, a review of a computer's speed may be "very good" but several months later the speed is only average.

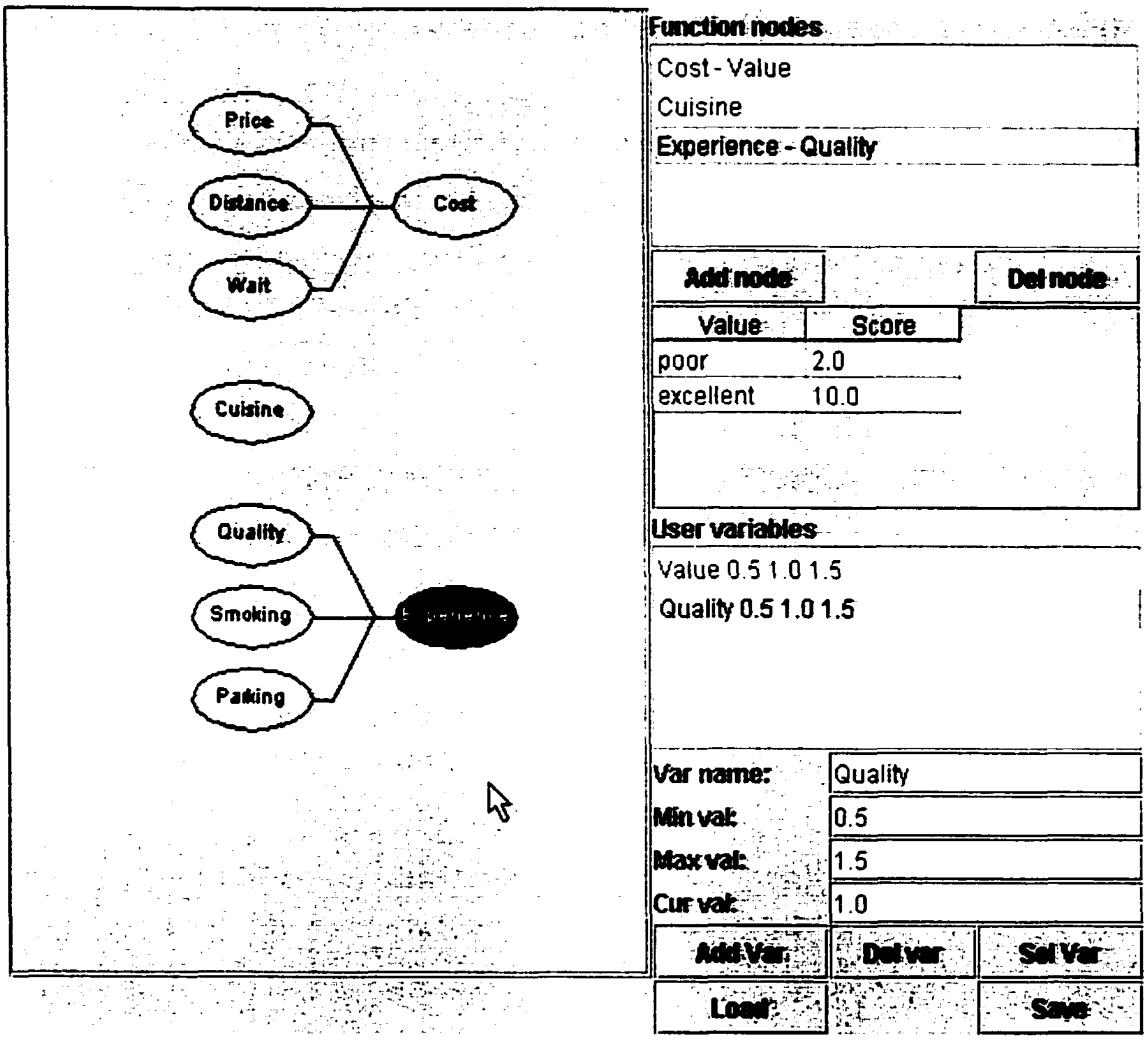
One advantage to using evidence nodes is that an uninstantiated evidence node will have no effect on the probability distribution of an instance. Thus adding evidence nodes dynamically does not change the probability distribution of the belief network. This outcome is a direct result of the belief network propagation algorithm described in section 15.3 of [62]. For a variable node N, the evidence nodes are child nodes that influence the node through *evidentiary support*. In the case where the child node of a node is uninstantiated and has no other children, then the evidentiary support from that child node is a uniform distribution, which has no effect on the probability distribution of the parent node. By definition, evidence nodes are child nodes of the variable node they influence with no other connections to the belief network. Thus, any uninstantiated evidence node has no effect on the probability distribution of the variable node. The feature nodes of the network have the same probability distribution if all of the evidence nodes that could possibly be added to the network were included. This "implicit" belief network would take a great deal of time to calculate because of the huge increase in the number of nodes, but the feature nodes, which are used by the utility function, would have the exact same probability distribution as our "explicit" network, which only includes the evidence nodes that are instantiated[4].

3.1.4 Utility score

The second part of each instance in **Figure 3.2** is the utility score. The utility score for a particular instance takes as input the probability distribution of the belief network and returns the expected utility for selecting that object from the set of possible objects. The utility score can be any function that takes as input the probability distributions of the variable nodes in the belief network. In many instances, it is also desirable for the utility scoring function to incorporate a set of *user variables* that are set by the user at the beginning of the gathering session. The user variables take into account individual user preferences that may vary from person to person. For example, in a digital camera purchasing decision, one person may value picture quality more than price or vice-versa. The expert designing the utility scoring function can add user variables to the utility function to consider these preferences as well. In our prototype systems, the user is presented with a set of controls at the beginning of the gathering session to specify their individual preferences. **Figure 3.10** shows the utility scoring function for the restaurant decision model used in section 6.6.

Definition 5 *The utility score for a belief network is defined as* $Us(B,V)$ *where* B *is the probability distribution of a subset of nodes in the belief network for the instance and* V *are the user variables used to describe the individual preferences of the user.*

[4] The rho/lambda propagation algorithm presented in [12] demonstrates how an uninstantiated child node with only one parent will not influence the parent node. The lambda message that it sends does not influence the probability distribution of the parent. Thus a uninstanted evidence node affects the influence diagram as a node that does not exist. This is exactly the behavior that we want from a class of nodes that model evidence.



Function nodes

Cost - Value
Cuisine
Experience - Quality

Value	Score
poor	2.0
excellent	10.0

User variables

Value 0.5 1.0 1.5
Quality 0.5 1.0 1.5

Var name: Quality
Min val: 0.5
Max val: 1.5
Cur val: 1.0

Figure 3.10 - Restaurant utility function

In our prototype system, the utility scoring function is calculated by assigning a score to every state for a subset of variable nodes in the belief network. In **Figure 3.10**, we have assigned utility scores for the states of the nodes *Cost*, *Cuisine*, and *Experience*. For the *Experience* variable node, the two values are: *Poor* = 2.0 and *Excellent* = 10.0. The *Experience* variable node also has a user variable called *Quality* (not to be confused with the variable node *Quality*). The user sets this variable at the beginning of the gathering session[5] to describe how important the *Experience* variable node is to this particular user. To calculate the utility score for this variable node the system takes the sum of multiplying each state value by the probability of that state being true and then multiplies this value by the user variable.

$$Us(B,V) = \sum_{f_i \in B} \left(\sum_k \Pr(f_i = s_{i,k}) Uf(i,k) \right) Vf(i) \tag{3.1}$$

Where B is the set of features f_i that influence the utility score
V is the set of user variables $Vf(i)$ that influence the utility score
f_i is the feature
$s_{i,k}$ are the states of f_i
$\Pr(f_i = s_{i,k})$ is the probability that feature f_i is state $s_{i,k}$
$Uf(i,k)$ is the utility score for feature f_i being state $s_{i,k}$
$Vf(i)$ is the user variable multiplier for feature f_i

For example if the probability distribution for the *Experience* variable node was Pr(Poor) = 0.3 and Pr(Excellent) = 0.7, and the state values were Sv(Poor) = 2.0 and Sv(Excellent) = 10.0, and the user variable *Quality* was 1.2, then the utility score for the *Experience* variable node would be (0.3 x 2.0 + 0.7 x 10.0) x 1.2 = 9.12. The utility score for each variable node in the subset specified by the utility scoring function is calculated and the sum is the utility score for the belief network.

[5] The experimental systems described in this dissertation assign the values of the user variables at the beginning of the gathering session and do not change them over the course of the session. There is no reason why the user variable could not be adjusted as the gathering session was in progress. Changing the user variables mid-session would have the effect of altering future queries made by the system to maximize the new selection criteria.

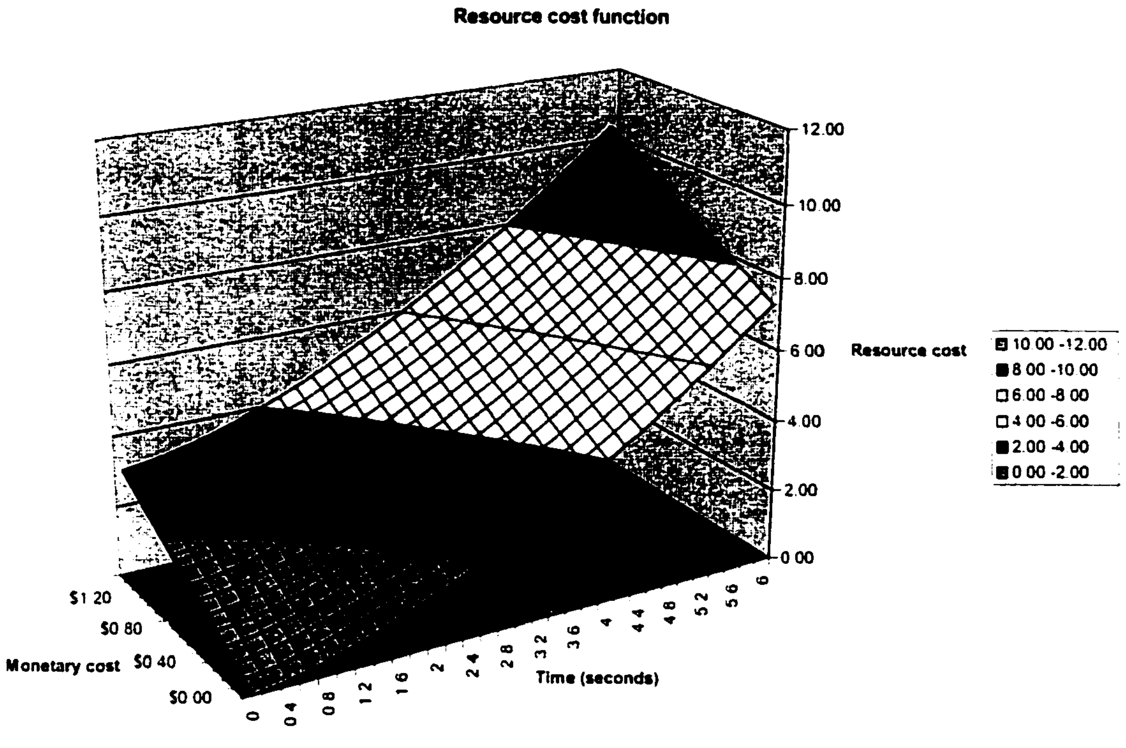


Figure 3.11 - Sample cost of time function with a linear monetary cost and an exponential time cost

3.2 Cost function

The *cost function* represents the cost in "utils" of spending time and resources during a gathering session. **Figure 3.11**, **Figure 3.12** and **Figure 3.13** show three sample resource cost functions. **Figure 3.14** shows the tool used by the final prototype system to define a resource cost function. Resource cost functions can be defined as any non-decreasing function in regards to every input variable. This means that for any input variable used to generate the resource cost (in the experimental system, the input variables for the resource cost function are: the length in time of the gathering session and the money spent during the session), if input variable increases, then so does the resource cost.

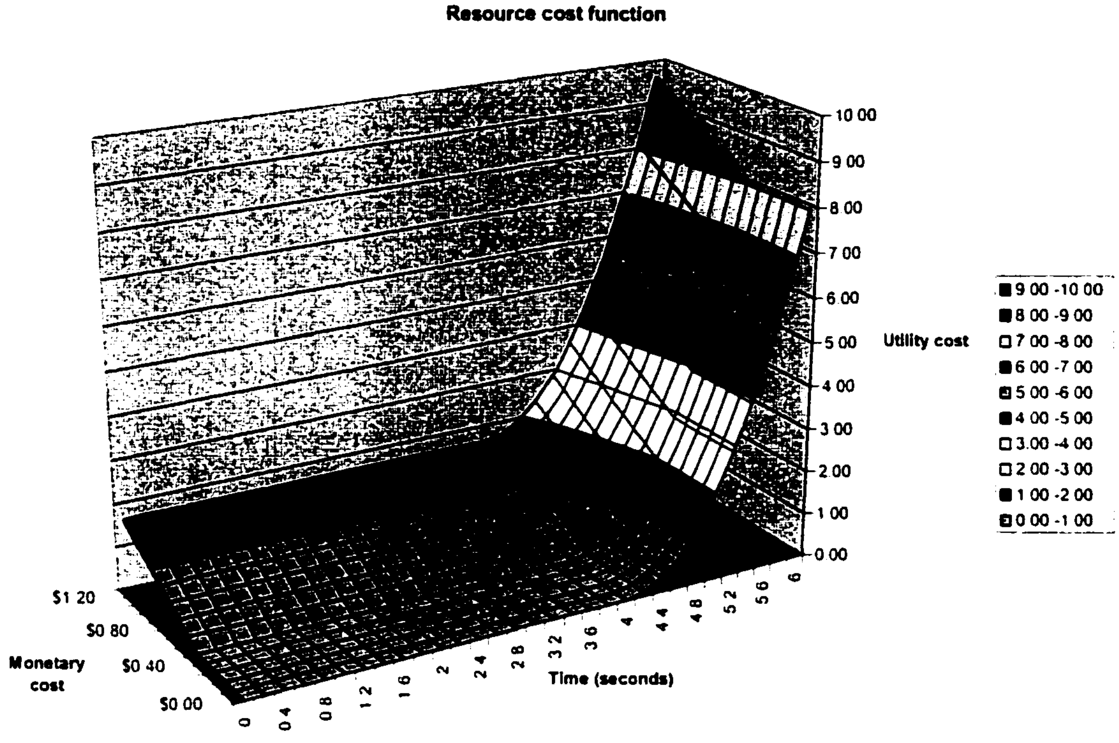


Figure 3.12 – Sample cost of time functions with a "grace" quantity of time and money, which the gathering session may use with no resource cost

Definition 6 *The resource cost function is defined as $C(t,m)$ where t is the time spent gathering information and m is the money spent on the queries issued during the gathering session.*

Figure 3.11 shows a sample resource cost function. In this example, the resource cost increases exponentially as the amount of time used during the gathering session increases. The resource cost also increases linearly as money is spent during the gathering session. This type of resource cost function represents a user who values both the money spent for the decision as well as the time used. The user is not willing to give the system any grace amount of time or money to use for the decision.

Figure 3.12shows a sample resource cost function in which the user has specified that there is no cost for the system to spend up to $0.50 for the search and to take up to 4

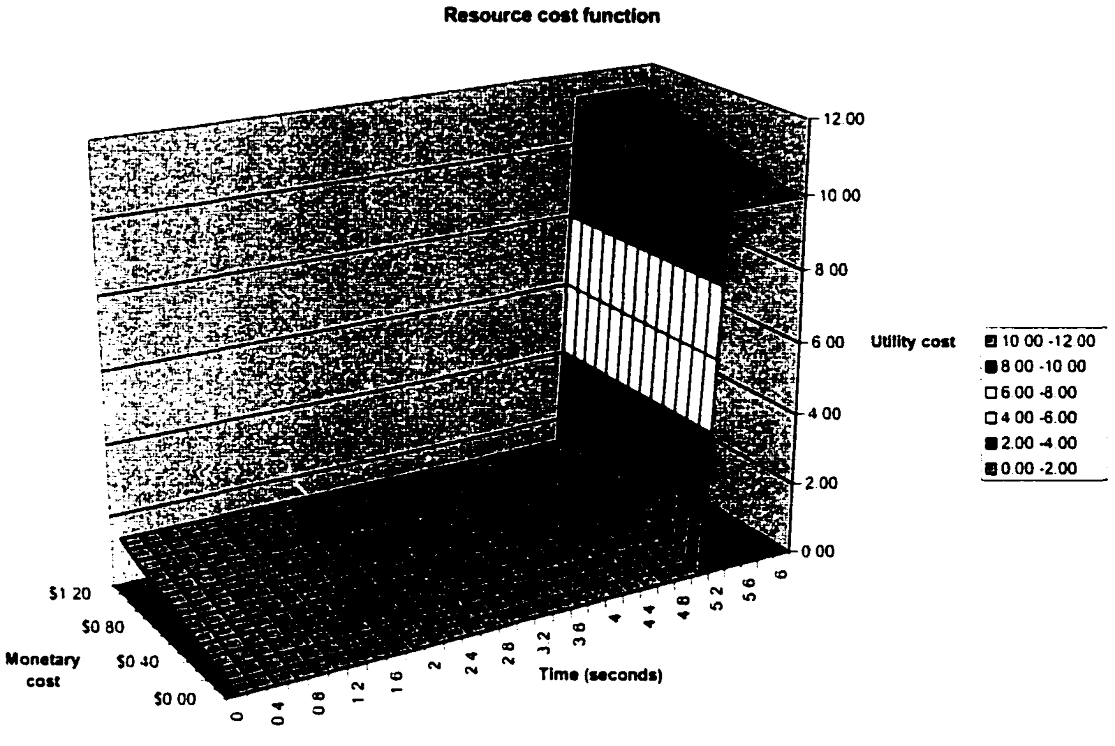
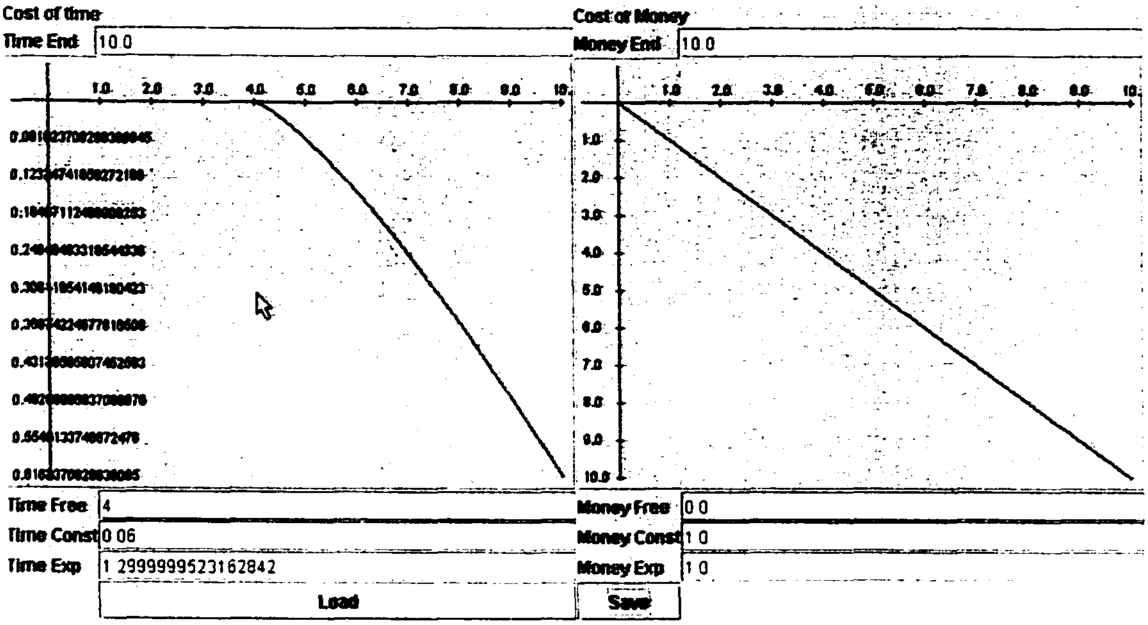


Figure 3.13 – Sample cost function with a deadline time

seconds for the search. Money spent beyond the "grace" amount has a linear resource cost and time spent beyond the "grace" amount has an exponential resource cost.

Figure 3.13 shows a sample resource cost function with a "deadline" time. Up until five seconds have passed, there is no cost for the system to spend more time continuing the search. After five seconds have passed in the information gathering session, the cost of using more time increases dramatically. In this example, the resource cost of using monetary resources in linear and has no grace amount of money that may be spent by the system during the information gathering session.

Figure 3.14 shows how the resource cost function is specified by the prototype value-driven information gathering system described in section 6.5 and 6.6. The user specifies a resource cost function separately for the time and money. The utility resource cost calculated from each function is then added together to calculate the total resource cost. The cost function used by the prototype system is described in equation



Cost of time

Time End 10 0

Time Free 4

Time Const 0 06

Time Exp 1 2999999523162842

Cost of Money

Money End 10 0

Money Free 0 0

Money Const 1 0

Money Exp 1 0

Figure 3.14 – Restaurant resource cost function

(3.2). Of course, the system could use any other type of resource cost function, but this system was flexible enough for the experimental requirements.

$$C(t,m) = k_t(\min(t - g_t, 0)^{e_t}) + k_m(\min(m - g_m, 0)^{e_m}) \qquad (3.2)$$

Where

$C(t,m)$	is the resource cost function for the prototype value-driven information gathering system
k_t, k_m	is the constant multiplier for the cost of the resource (time or money)
e_t, e_m	is the exponential factor applied to the cost of the resource (time or money)
g_t, g_m	is the grace amount of the resource which the system may use at no cost (time or money)

3.3 Information sources

Information sources are external processes that can be queried by a value-driven information gathering system. Value-driven information gathering systems have no control over when this information returns, only over when the information source is queried. The integral of the response-function does not necessarily have to sum to 1.0, since some information sources may have a chance of failing and not returning evidence at any point in the future.

Definition 7 *Information from a source (s) is defined as* $S_s = (F_s, \Pr_s(t), \beta_s, M_s)$, *where* F_s *is the set of variable nodes (for a particular instance) that the source returns information about,* $\Pr_s(t)$ *is the probability of the source returning information at any time after being queried,* β_s *is the reliability of the information source and* M_s *is the monetary cost of querying the information source.*

It is important to note that value-driven information gathering places no restrictions on the shape of the response-function. This means that any information source can be modeled for use by a value-driven system, be it a query to a web page, a SQL inquiry to a database, an I/O operation on a hard disk, or querying a sensor in the outside world. In general, many systems have simply ignored the probabilistic response of information sources, or have used some form of guaranteed response time.

This dissertation focuses on using the Internet as a test-bed for value-driven information gathering. Information sources on the Internet follow this model very well. They have different response-functions based on the speed of their server, their physical location in the world and the general load on the Internet at the time of the query. Fortunately, it is possible to learn a good approximation of the response probability for an information source on the Internet. It is important to note that when we talk about an

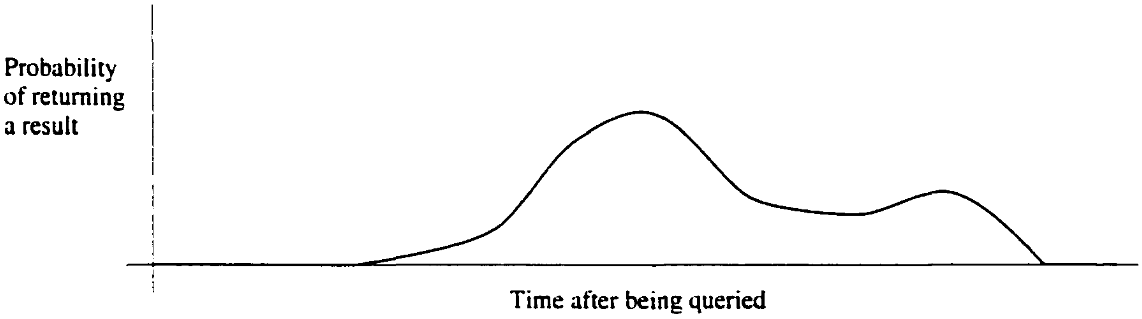
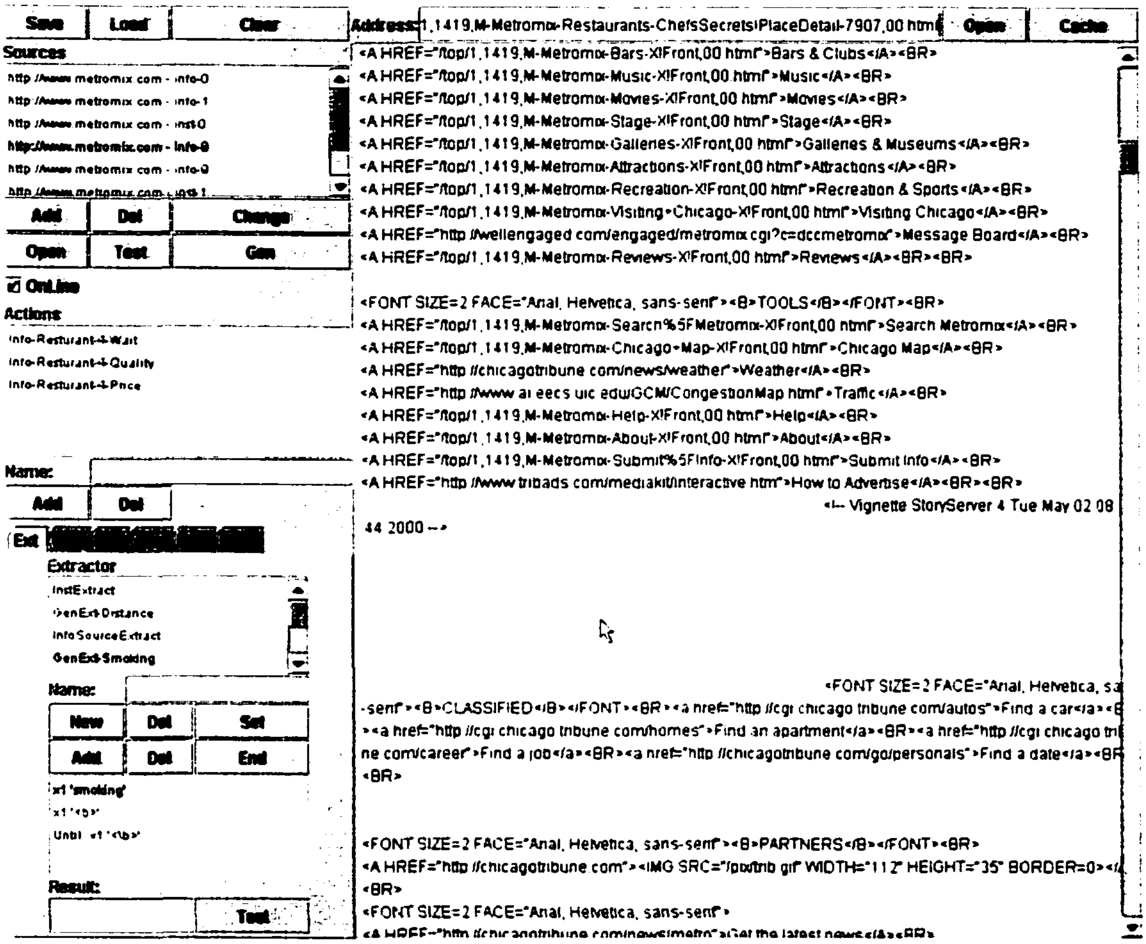


Figure 3.15 - Information source response-function used in value-driven information gathering



Address: 1,1419,M-Metromix-Restaurants-ChefsSecrets!PlaceDetail-7907,00 html

Sources:

```
http //www metromix com - info-0
http //www metromix com - info-1
http //www metromix com - inst0
http://www.metromix.com - Info-8
http //www metromix com - info-0
http //www metromix com - inst 1
```

Actions:

```
Info-Resturant-+-Wait
Info-Resturant-+-Quality
Info-Resturant-+-Price
```

Extractor:

```
InstExtract
GenExt-Distance
InfoSourceExtract
GenExt-Smoking
```

```
xt 'smoking'
x1 '<b>'
Until x1 '<\b>'
```

```
<A HREF="/top/1,1419,M-Metromix-Bars-X!Front,00 html">Bars & Clubs</A><BR>
<A HREF="/top/1,1419,M-Metromix-Music-X!Front,00 html">Music</A><BR>
<A HREF="/top/1,1419,M-Metromix-Movies-X!Front,00 html">Movies</A><BR>
<A HREF="/top/1,1419,M-Metromix-Stage-X!Front,00 html">Stage</A><BR>
<A HREF="/top/1,1419,M-Metromix-Galleries-X!Front,00 html">Galleries & Museums</A><BR>
<A HREF="/top/1,1419,M-Metromix-Attractions-X!Front,00 html">Attractions</A><BR>
<A HREF="/top/1,1419,M-Metromix-Recreation-X!Front,00 html">Recreation & Sports</A><BR>
<A HREF="/top/1,1419,M-Metromix-Visiting+Chicago-X!Front,00 html">Visiting Chicago</A><BR>
<A HREF="http //wellengaged com/engaged/metromix cgi?c=dccmetromix">Message Board</A><BR>
<A HREF="/top/1,1419,M-Metromix-Reviews-X!Front,00 html">Reviews</A><BR><BR>

<FONT SIZE=2 FACE="Arial, Helvetica, sans-serif"><B>TOOLS</B></FONT><BR>
<A HREF="/top/1,1419,M-Metromix-Search%5FMetromix-X!Front,00 html">Search Metromix</A><BR>
<A HREF="/top/1,1419,M-Metromix-Chicago+Map-X!Front,00 html">Chicago Map</A><BR>
<A HREF="http //chicagotribune com/news/weather">Weather</A><BR>
<A HREF="http //www ai eecs uic edu/GCM/CongestionMap html">Traffic</A><BR>
<A HREF="/top/1,1419,M-Metromix-Help-X!Front,00 html">Help</A><BR>
<A HREF="/top/1,1419,M-Metromix-About-X!Front,00 html">About</A><BR>
<A HREF="/top/1,1419,M-Metromix-Submit%5FInfo-X!Front,00 html">Submit Info</A><BR>
<A HREF="http //www tribads com/mediakit/interactive htm">How to Advertise</A><BR><BR>
                                                   <!-- Vignette StoryServer 4 Tue May 02 08
44 2000 -->



                                                   <FONT SIZE=2 FACE="Arial, Helvetica, sa
-serif"><B>CLASSIFIED</B></FONT><BR><a href="http //cgi chicago tribune com/autos">Find a car</a><B
><a href="http //cgi chicago tribune com/homes">Find an apartment</a><BR><a href="http //cgi chicago tri
ne com/career">Find a job</a><BR><a href="http //chicagotribune com/go/personals">Find a date</a><BR
<BR>


<FONT SIZE=2 FACE="Arial, Helvetica, sans-serif"><B>PARTNERS</B></FONT><BR>
<A HREF="http //chicagotribune com"><IMG SRC="/pix/trib gif" WIDTH="112" HEIGHT="35" BORDER=0></
<BR>
<FONT SIZE=2 FACE="Arial, Helvetica, sans-serif">
<A HREF="http //chicagotribune com/news/metro">Get the latest news</A><BR>
```

Figure 3.16 - Restaurant information sources database

information source on the Internet, we are not talking about a server, but a particular web page. Each web page is viewed as an information source in this model. When it is necessary to talk about a collection of information sources on the Internet owned by the same organization, we will use the term *site*.

Figure 3.16 shows the information sources database for the restaurant example. The table on the top left of the figure shows the total list of sources accessible for this decision. The large window on the right half of the figure shows the raw HTML text for one of the information sources which can be accessed by the system. The table on the middle left of the figure shows a list of extraction actions that can be performed on this information source. They possible actions are: Extract the wait time for this restaurant, extract the quality of the food for this restaurant, and extract the price of a meal at this restaurant. The table at the bottom left shows the construction of an extractor for determining if the restaurant is non-smoking or not.

3.4 Gathering session

A *gathering session* is the period of time starting with the launch of the value-driven information gathering system and ending when the system returns a decision. The value-driven information gathering system determines when it will return a decision based on the expected future utility of the gathering session. **Figure 3.17** shows how the expected utility curve changes over the course of the session as queries are made,

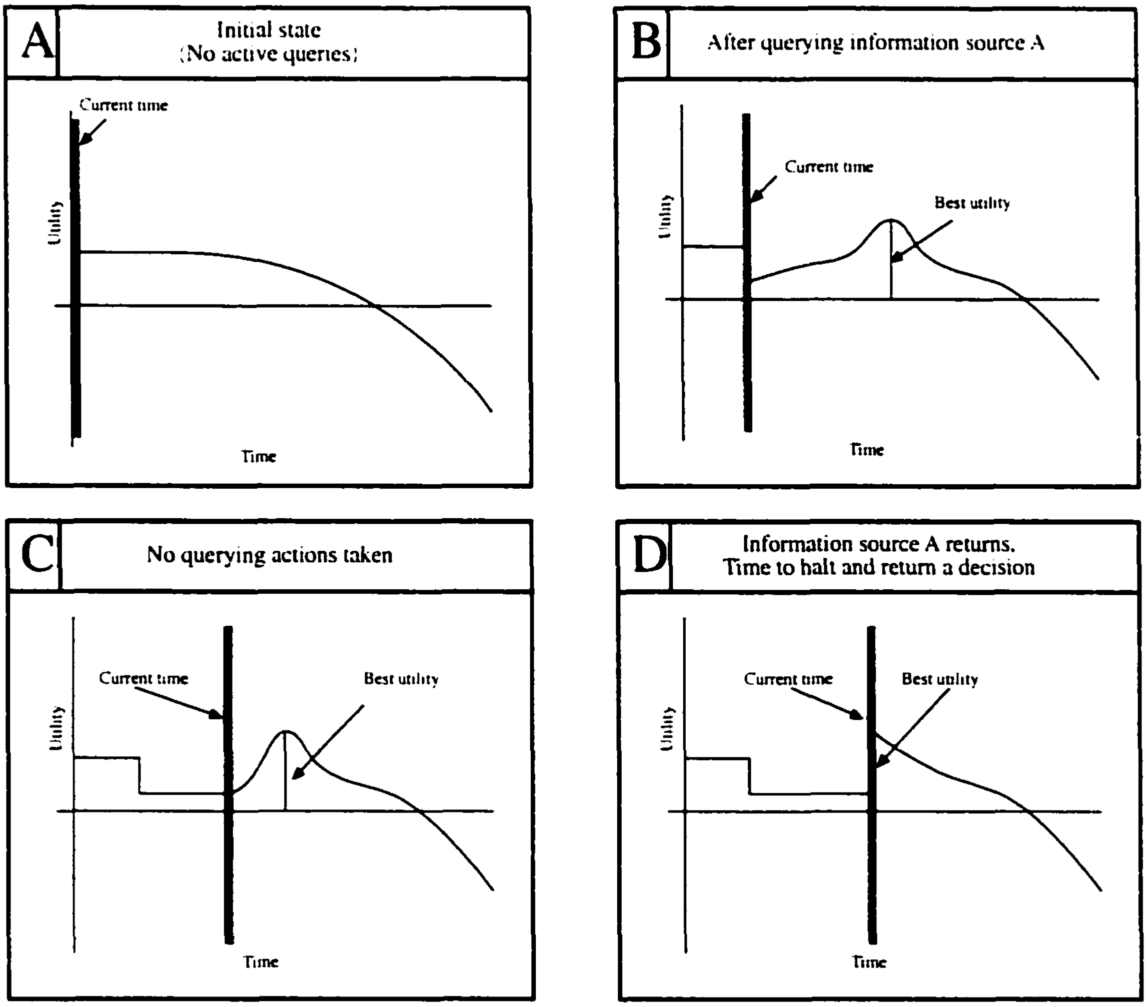


Figure 3.17 - Expected utility curve over time

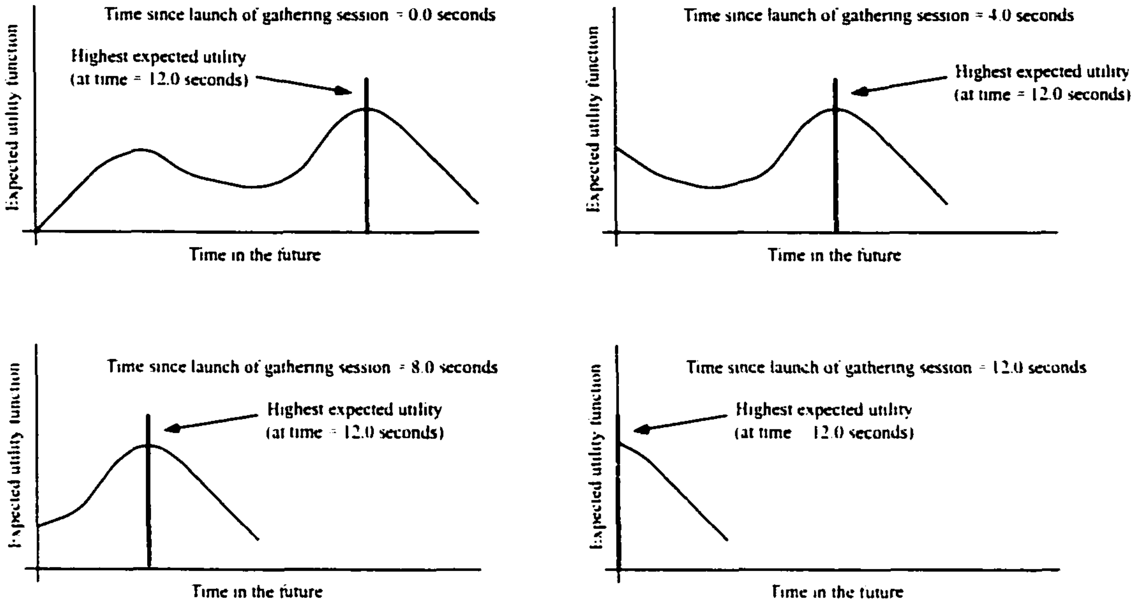


Figure 3.18 – The expected utility function over the course of the gathering session (no additional queries made beyond the initial set at time zero)

evidence is returned and time and resources are used. This set of graphs are explained in more detail in section 4.1 where the value-driven information gathering algorithm is described in detail.

Definition 8 *The expected utility function is defined by $Ue(t \mid R,Q)$, where t is the relative time, R is the set of responses from information sources that the value-driven system has received, and Q is the set of active queries (queries that have been made but have not yet been returned by the information source).*

The expected utility function assumes that no future queries will be made by the value-driven system. The value-driven system halts the gathering process when the expected utility function does not improve in the future. **Figure 3.18** shows an example of what the expected utility function looks like over the course of a simplified gathering session. In order to simplify the figure, this expected utility function is for a gathering session in which no queries are made after the initial set of queries are made at time zero. The gathering session continues until the expected utility function at time zero (relative to the time since the launch of the gathering session) is the maximum value of the expected utility function. In this example, the maximum expected utility function score occurs 12 seconds after the launch of the gathering session. Since the system is not adding any additional queries in this example, the shape of the expected utility function does not change over time[6]. As the gathering session continues, the system continues to evaluate the expected utility function and decide if the expected utility function at the current time is the maximum of the expected utility function. At this point, continuing the gathering session will only reduce the expected utility of the system. Once this maximum has been reached, the system halts the information gathering process and returns the current best decision. In practice, this usually occurs once an information source returns new evidence, which, once incorporated into the decision model, can radically change the expected utility function. This can make the value of the expected utility function at time zero the maximum value.

One final note on implementation, the decision to halt the gathering session is done after the expected utility function is evaluated for all potential queries and the best query has been made and added to the query pool. There are situations when an information source has returned data, the expected utility function is at its maximum at time zero, and after a new query has been made, the expected utility function maximum value is again at some point in the future.

[6] Actually, even in this situation the expected utility function will change shape over time. There are two reasons for this: The expected utility function will change shape when any of the queries return information. Moreover, the expected utility function will change shape even if no information is returned because the probability of the information sources returning at a given time changes as the time passes.

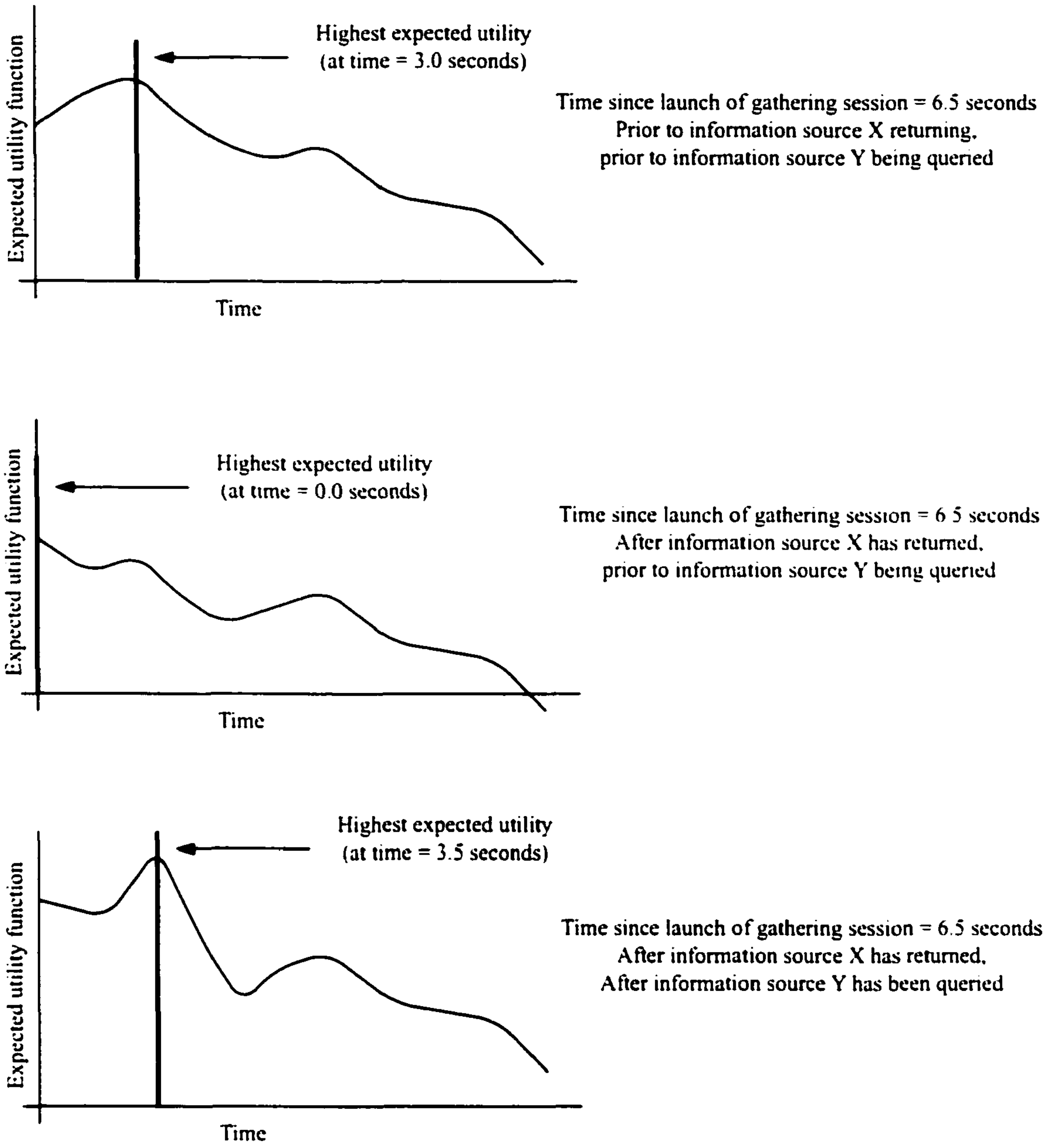


Figure 3.19 – Expected utility function during a query returning and an new query being made

Figure 3.19 shows an example of this situation. In the figure, 6.5 seconds have passed in the information gathering session. The top graph shows the expected utility function evaluation before information source X has been incorporated into the decision model. The maximum value of the expected utility function is 3.0 seconds in the future, the information gathering process will continue if nothing changes. The second graph show the expected utility function after information source X has returned and the data has been incorporated into the decision model, but before the new information source Y, has been queried. At this point, the maximum value of the expected utility function is at time zero, if no new queries are made the information gathering process would halt and return a decision. The third graph shows the expected utility function after information source Y has been queried. The new query has altered the expected utility function so that the maximum value of the function is 3.5 seconds in the future. The information gathering process will continue.

3.4.1 Querying constraints

Querying constraints represent the maximum number of active queries that the system may have at any one time. If the system reaches this maximum, it may have to abandon a query before it returns a result in order to launch a new query that will improve the expected utility function more than the query being abandoned. In the first set of experiments, we constructed a system in which only one query could be active at any time (see section 6.2). The second set of experiments (described in section 6.3, 6.4 and 6.5) removes this restriction, and the final experimental system described in this dissertation (see section 6.6) allows the number of active queries to be set to n at the beginning of the gathering session.

3.4.2 The value-driven information gathering problem

The value-drive information gathering problem is: which information sources should the system query in order to maximize the expected quality of the decision, taking into account the cost and benefits of gathering that information. Determining which sources to query is based on how reliable the information is, how it will affect the decision model and how long it will take for the information to return. The next chapter will describe in detail how we compute the *value of a query*, which the system uses to determine which query to make.

Our basic approach is to deal with the problem myopically and not try to develop a gathering plan ahead of time that uses multiple queries in conjunction. The value-driven information gathering system uses a myopic approach because of the high variability in the time for information sources to return and the large effect that unexpected responses can have on future querying priorities. Discovering that one instance has a highly desirable feature can completely change the optimal querying plan. Our approach is to use the current state of the system, which includes the information that has been returned and the queries that are still pending to score every potential querying action and select the query that will increase the expect utility function the most.

CHAPTER 4

VALUE-DRIVEN INFORMATION GATHERING

This chapter outlines the central problem facing a value-driven information gathering system: Given a decision model, a set of information sources, and a set of resource constraints, a value-driven information gathering system should query information sources in order to maximize decision quality. Because of the high variability of when information sources return and the effect of their values on the system, the system uses a myopic strategy for planning the information gathering session. This simplifies our task to determining which single query will have the greatest effect on increasing the quality of our decision. We can thus assign to every potential information source *the value of a query*. This value is the expected increase in the utility function from querying the information source.

Determining the value of a query in value-driven information gathering depends on four factors: the set of knowledge already acquired by the system, the set of active queries that may return information in the future, the characteristics of the potential information sources and the resources already used by the system. The first section in this chapter describes the algorithm for value driven information gathering. Section 4.2 defines the equations for determining the *value of a query*. The value of a query is the "score" which is used for sorting potential queries and determining which query to make next. Section 4.3 discusses a method for evaluating the expected increase in utility from querying an instance source for new instances to add to the decision model. In addition, section 4.4 describes situations in which the number of outstanding queries is limited. If this is the case, then there are circumstances in which it is advantageous to halt queries before they have returned so that a new query can be made[7].

[7] In the case of an information gathering system that operates on the Internet, halting a query means closing the socket that monitors the connection for return data and halting the computational process that will extract the data from the raw information returned

```
1. Initialize the utility curve.
   Repeat
2. Calculate the value of querying each information source relevant to the decision.
3. Calculate the cost-adjusted increase to the utility curve. This is the value of the
   query.
4. If any query increases the overall expected utility, add the query, which increases
   the expected utility the most.
   Until the expected utility curve is currently at its maximum
```

Figure 4.1 - The VDIG algorithm

4.1 Value-Driven Information Gathering Algorithm

Figure 4.1 shows the algorithm used for determining which information source (if any) to query over the course of the information gathering session.

Throughout this chapter, we will be discussing the *utility curve*. The utility curve is a graph that represents both the actual utility of the value-driven process and the future expected utility. **Figure 4.2** shows the utility curve of a value-driven information gathering session at various points in time.

In **Figure 4.2** the portion of the curve to the left of the current time is the actual utility. The actual utility is based on what information has been retrieved and the resources spent by the system. The portion to the right of the current time bar represents the value-driven systems expected utility in the future of the gathering session. This assumes that the system will make no further queries. These graphs also include the resource cost function at each point in time as well. In **Figure 4.2** (a) the gathering session has just begun. Not queries have been made and the curve to the right of the current time bar is the resource cost function. The utility curve is positive at this point because the system has a default action that it can perform with a positive utility score.

In **Figure 4.2** (b) the value-driven system has made a query. The utility curve at the current time has decreased because of the monetary cost of making the query. Making this query has also had the effect of adding a new expected increase in the utility at

by the query. Both of tasks use computational resources that might be better spent working on another query.

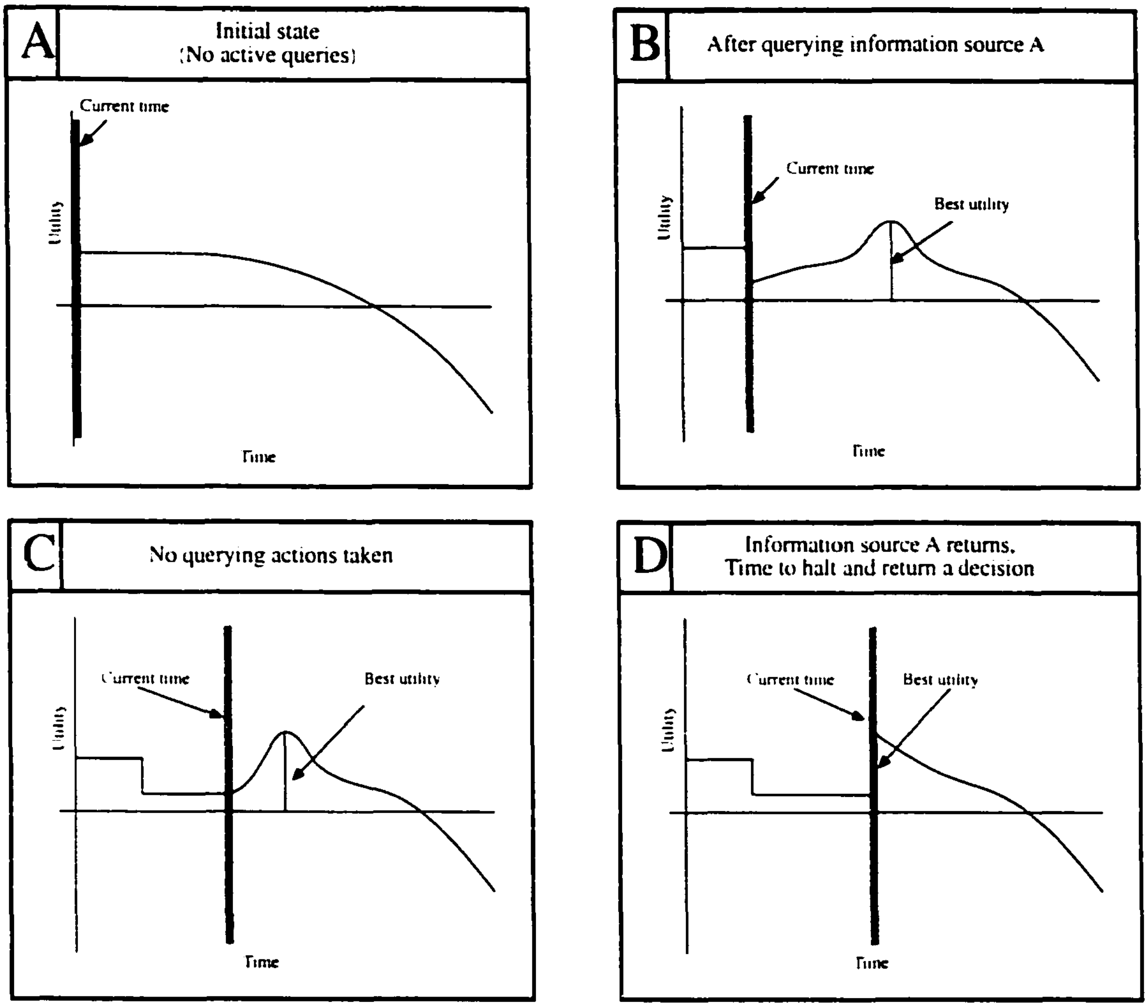


Figure 4.2 - Expected utility curve over time

some point in the future. The new utility curve to the right of the current time bar represents both the expected increase in the quality of the decision from the data that this query will return, but also the resource cost of waiting for that information to return. The value-driven system determines the time at which the system is expected to reach its "best utility" and continues the gathering session until the "best utility" time is the same as the current time.

Figure 4.2 (c) shows the utility curve several seconds latter. No new queries have been made and the query that was made in (b) has not yet returned. The utility curve to the left of the current time bar has remained flat because no new information has returned, the expect utility of the decision (minus the cost of making the query in (b)) has not changed. The utility curve to the right of the current time bar, however, has changed. This change comes from the change in the response-function for the query as time has passed (see equation (4.6) for more details on how the response changes as a function of elapse time since the query was made).

Finally, **Figure 4.2** (d) show the change in the utility curve once the query has returned information. The utility at the current time bar has spiked now that the system has concrete information to use to increase the expected quality of the decision. Assuming no new queries are going to be made, the gathering session has reached a maximum at this point and the most effective course of action is to use this information to return the best decision that the decision model can make before the system must pay for more resources (in this case, time). At this point, the "best utility" score is at the current time.

4.2 Determining the value of a query

The key to the value-driven information gathering algorithm is to determine the net value of a query. Once we have this function, we can evaluate potential queries and activate the one with the highest value. Throughout this discussion, we will be referring to three sets of queries: The set of queries that have returned evidence (R), the set of active queries that have been made but have not yet returned a value (Q), and the set of potential queries that may be made in the future (P). As the value-driven system executes a querying session, queries will move from P to Q and when those queries return they will move from Q to R.

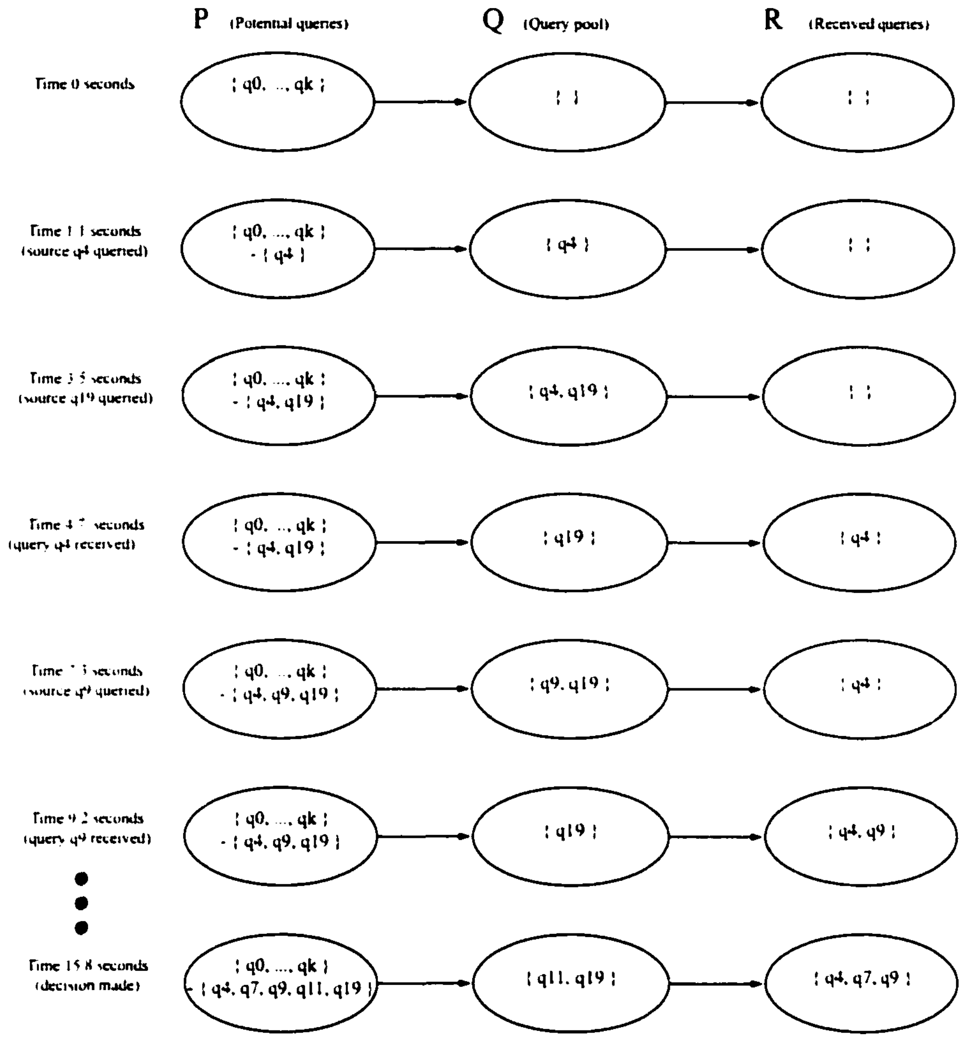

P (Potential queries)
Q (Query pool)
R (Received queries)
Time 0 seconds
{ q0, ..., qk }
{ }
{ }
Time 1.1 seconds (source q4 queried)
{ q0, ..., qk } - { q4 }
{ q4 }
{ }
Time 3.5 seconds (source q19 queried)
{ q0, ..., qk } - { q4, q19 }
{ q4, q19 }
{ }
Time 4.7 seconds (query q4 received)
{ q0, ..., qk } - { q4, q19 }
{ q19 }
{ q4 }
Time 7.3 seconds (source q9 queried)
{ q0, ..., qk } - { q4, q9, q19 }
{ q9, q19 }
{ q4 }
Time 9.2 seconds (query q9 received)
{ q0, ..., qk } - { q4, q9, q19 }
{ q19 }
{ q4, q9 }
Time 15.8 seconds (decision made)
{ q0, ..., qk } - { q4, q7, q9, q11, q19 }
{ q11, q19 }
{ q4, q7, q9 }

Figure 4.3 – The state of the potential query set (P), the query pool (Q) and the returned evidence (R) during a gathering session

The value of a query is defined as the difference in the maximum expected utility when a query is added to the pool of active queries. **Figure 4.2** (b) shows the "best utility" changing after the system has launched the query. Equation (4.1) defines the value of a query, q_i, based on the difference in the expected utility curve. The subscript i in q_i refers to the individual query being evaluated out of the set of all potential queries ($\{q_0, \ldots, q_k\}$.

$$\Delta(q_i) = \overline{U}(R,Q \cup \{q_i\}) - \overline{U}(R,Q) \tag{4.1}$$

Since the value-driven information gathering system controls the time at which the information gathering process halts, the expected utility of the information gathering session ($\overline{U}(R,Q)$) is the expected utility at the best stopping time for the expected utility function ($\overline{U}(t \mid R,Q)$). Equation (4.2) shows the expected value of the gathering session based on the expected utility curve.

$$\overline{U}(R,Q) = \arg\max_t U(t \mid R,Q) \tag{4.2}$$

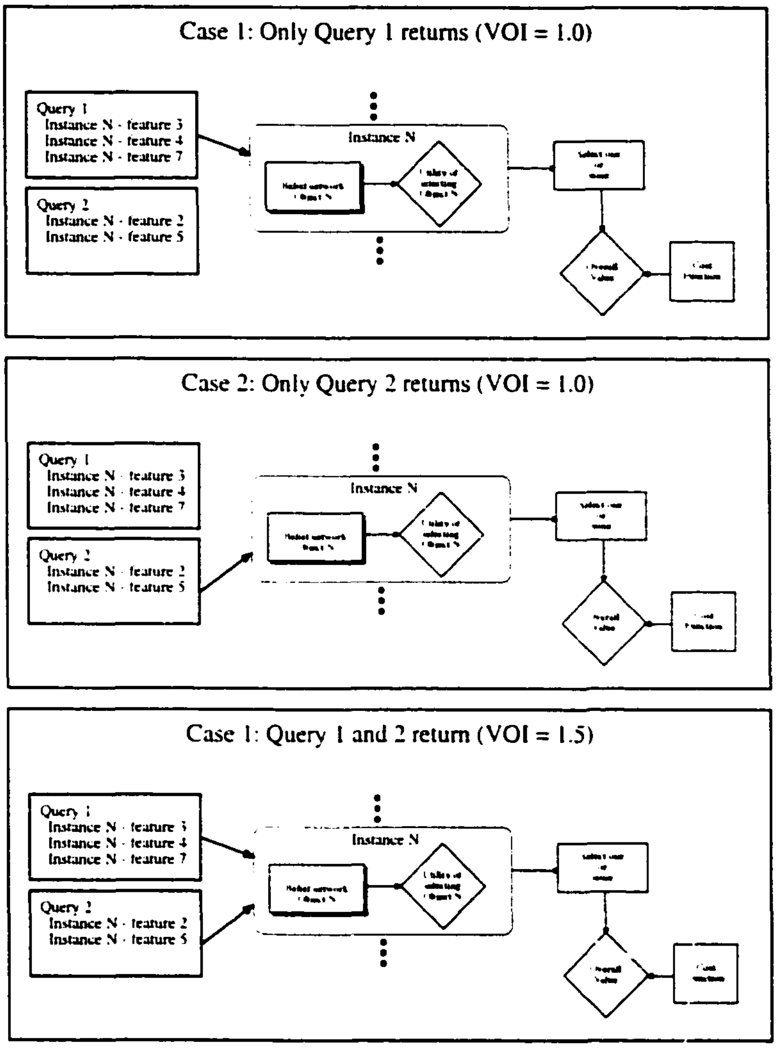
The expected utility function returns utility values as a function of time, assuming that *no future queries* are made. This function is based on the queries that have returned evidence (R), the queries that are active but have not yet returned (Q) and the cost function ($C(n+t, R \cup Q)$). Where n is the current time in the gathering process and t is the future time. The R and Q sets in $C(n+t, R \cup Q)$ represent the initial query cost for each query that has been made (whether it has returned, R, or is in the query pool, Q). In general, the initial querying cost is the monetary cost that the information source may impose to receive a query. Equation (4.3) defines the expected utility function for the gathering session.

$$U(t \mid R,Q) = \max_i U_i(t \mid R,Q) - C(n+t, R \cup Q) \tag{4.3}$$

$U(t \mid R,Q)$ is determined by calculating the expected utility for each instance ($U_i(t \mid R,Q)$) and taking the maximum, minus the cost function. The decision model used by value-driven information gathering systems selects the instance with the highest expected utility. The net effect of this is that queries that increase the expected utility of a particular instance can only change the overall expected utility of the system if the expected utility of that instance becomes greater than the expected utility of the best instance current known to the system. For example, if there are three instances with expected instance utilities of 3, 8, and 10, a query that increases the expected utility of the first instance from 3 to 9 still has an expected utility for the entire system of 0. Only increases that will make the expected utility of a particular instance greater then the expected utility of the best instance (in this case greater then 10) have any impact on expected utility of the system as a whole. Therefore, increasing the value of any particular instance will not affect the utility curve unless there is a chance that it will become the best instance, that is, the instance with the highest instance utility score). Equation (4.4) describes the utility for each specific instance in the decision model.

$$U_i(t \mid R,Q) = \sum_{\alpha \subseteq Q_i} \Pr(\alpha \mid t) V_i(R_i, \alpha) \tag{4.4}$$

To calculate the expected utility for a specific instance ($U_i(t \mid R,Q)$), the system must take the subset of active queries (Q) that return information about the instance i (Q_i is the subset of active queries that apply to instance i) and averages over all possible subsets (α) of evidence that may be returned by these queries at time t. In order to prevent the performance of the systems from degrading too greatly during this computation, the system limits the maximum number of queries that are evaluated during any individual time step. This insures that the number of subsets of evidence evaluated has an upper bound. The system must calculate each subset of evidence from the information source because the value of information is not additive for sets of information sources. **Figure 4.4** shows the value of information for the decision model for two active queries. Fortunately, we do not need to consider every piece of evidence, because if an information source returns, all of its evidence is returned. So the number



Case 1: Only Query 1 returns (VOI = 1.0)

Case 2: Only Query 2 returns (VOI = 1.0)

Case 1: Query 1 and 2 return (VOI = 1.5)

Figure 4.4 - The value of information for multiple information sources may not be additive

of states that must be evaluated is based on the number of outstanding queries, not the number of outstanding pieces of evidence[8].

Calculating the probability of a subset of queries returning at any given time given a specific querying pool is calculated by multiplying the probability of each query, q, in

[8] The number of pieces of evidence is always greater than the number of active queries, because some information source may return several pieces of evidence when they return.

α returning at time t by the probability of all of the queries not in α returning at time t. Equation (4.5) defines this formula.

$$\Pr(\alpha \mid t) = \prod_{q \in \alpha} \Pr(q \mid t) \prod_{q \notin \alpha} (1 - \Pr(q \mid t)) \tag{4.5}$$

The value of $\Pr(q \mid t)$ is the expected response of the information source. The expected response will change based on the response-function for the information source as well as the how long ago the query was made. Equation (4.6) shows how, using Bayes rule, the expected response changes over time for an information source. The variable a is the time at which the query was initially activated and $\Pr_s(i)$ is the probability of the information source returning at time i, where $i = 0$ is the time at which the query was launched (this is the response-function, see section 0). The value of $\Pr_s(i)$ is collected from experience querying the system and other factors, such as network load and site traffic.

$$\Pr(q \mid t) = \frac{\Pr_s(t-a)}{1 - \sum_{i=0}^{t-a-1} \Pr_s(i)} \tag{4.6}$$

For example, assume that an information source exists with a response-function of $\Pr_s(0) = 0.3$, $\Pr_s(1) = 0.2$, $\Pr_s(2) = 0.1$, $\Pr_s(3) = 0.4$. If this query were launched at $t = 8$, then the probability of the source returning at each time step from $t = 8$ to $t = 11$ would be:

$$\Pr(q \mid 8) = \frac{\Pr_s(0)}{1} = \frac{0.3}{1} = 0.3$$

$$\Pr(q \mid 9) = \frac{\Pr_s(1)}{1 - \Pr_s(0)} = \frac{0.2}{1 - 0.3} = 0.286$$

$$\Pr(q \mid 10) = \frac{\Pr_s(2)}{1 - \Pr_s(0) - \Pr_s(1)} = \frac{0.1}{1 - 0.3 - 0.2} = 0.2$$

$$\Pr(q \mid 11) = \frac{\Pr_s(3)}{1 - \Pr_s(0) - \Pr_s(1) - \Pr_s(2)} = \frac{0.4}{1 - 0.3 - 0.2 - 0.1} = 1$$

The second component of equation (4.4) ($V_t(R_t, \alpha)$) is the expected value of the information given the evidence that has already been returned and the evidence returned by the queries in α. Section 3.1.3 describes how to incorporate evidence into the decision model.

4.2.1 Optimizations to calculating the value of a query

Both components in equation (4.4) can be optimized to reduce the computational demands required to calculate the value of a query.

The first component, $\Pr(\alpha \mid t)$, which represents the likelihood of a set of information sources returning information at time t, can be cached and only needs to be recalculated when a new information source for that instance is queried. As time progresses, equation (4.6) can be iteratively recalculated based on the previous value and the value of $\Pr_s(t)$.

The system can further control the computational resources required by $V_t(R_t, \alpha)$ and $\Pr(\alpha \mid t)$ by restricting the maximum number of queries that the subset α may contain. The queries in α, which are considered at any point in time t for calculating $V_t(R_t, \alpha)$, may also be sorted by their probability of returning at time t and the system can limit the number of queries that are evaluated to the k queries that are most likely to return during that time slice. Thus, instead of calculating $V_t(R_t, \alpha)$ and $\Pr(\alpha \mid t)$ for every combination of outstanding queries, we can reduce the number of calculations but limiting the combinations to the k queries most likely to return during that time slice. In reality, the probability of two or more queries returning during the same time slice is very low.

In the most restrictive case, the system can assume that the probability of multiple information sources returning evidence during the same time slice is zero. In cases where the polling time (how often the system check the queries to see if any have returned) is much shorter then the average time for the query to return, the probability of two queries returning simultaneously is very small.

The second component, $V_i(R_i, \alpha)$, is the value of information for the evidence returned by the set of queries α. This information may also be cached for any instance not affected by the query. This is because each instance only connects to the other instances through the utility node. Thus, the value of information for a set of evidence affecting one instance is independent and does not alter the value of information for any other instance.

Finally, in the case of decision models that are substantially larger than those used in this dissertation, the computational requirements of propagating the decision model can be further reduced through several approximation methods. Horvitz [42] describes a variety of optimization methods that can drastically reduce the average computational resources required to calculate the value of information for an influence diagram.

4.3 Determining the value of finding a new instance

In some cases, value-driven information gathering systems may have an information source available to them that can be queried to return new instances to add to the decision model. These types of data sources are referred to as *instance sources*. There are several approaches to incorporate instance sources into a value-driven system.

Determining when to add a new instance to the decision model is relatively straight-forward operation. The value-driven system maintains an empty instance that is evaluated just like any other instance. This instance differs only in two ways: 1) any query that can be made of an instance can be made to the blank instance, 2) the time required to query the instance sources is added to the response histogram for the query. If the value-driven system calculates that the best query is one of the queries that affect the blank instance, then the instance source is queried and the query is made on the instance that is returned.

Calculating the value of a new instance allows the system to expand its decision model dynamically until it is satisfied that it has a good enough sample to find a high-quality instance to recommend to the user. In general, this calculation matches what common sense would expect: The value of adding a new instance to the decision model is based on the resources left and the quality of the instances already in the decision model. In general, a value-driven system will add new instances at two points in time. The system will query for new instances at the beginning of the gathering session when variability is at its greatest and the system still has a large amount of resources. The system will also query for new instances when it has determined that none of the active instances are likely to have a high expected utility.

4.4 Limited number of outstanding queries

Value-driven systems may operate in environments in which a limited number of active queries are allowed. Although this case does not often occur in information gathering systems that operate on the Internet, in other domains it is possible that the system will have a limited number of outstanding queries that can be made at any one time[9]. For example, an information gathering system that operated with hardware system as the query mechanism, it would have a limited number of outstanding queries at any one time. In the case where the number of outstanding queries is limited, the value-driven system may have to decide whether to postpone making a new query or to halt a query that has not yet returned and replace it with the new query. In order to determine the impact of replacing one query with another, the *ongoing value of a query* must be calculated. The calculation of the value of a query is also made more complicated by the fact that the value of the new query may change depending on which

[9] Multiple user systems that operate on the Internet may also have a limit on the number of outstanding queries that the system can monitor. Maintaining and monitoring a socket connect to an internet server does have some computational cost, and a value-driven information gathering system that operated as a several for many users at once would have to take these computational resource costs into account.

query it is replacing, this function is the *replacement value of a query*. This section describes how the ongoing and replacement value of a query are calculated.

Evaluating the *ongoing value of a query* is very similar to calculating the value of a query as described in section 4.2. There are three main distinctions:

1. The responsiveness function has changed because the query was launched sometime in the past (the new responsiveness function can be calculated using equation (4.6). This equation describes how to adjust the response-function given the time the query was made and the original response-function for the information source).
2. The value of information for the evidence returned by the information source may change because of new information that has returned since the query was made.
3. The value of the query no longer has an initial querying cost (see $C(n+t, R \cup Q)$ in equation (4.3)) because the query has already been made and the initial querying cost has already been paid.

With these modifications, we can calculate the ongoing value of a query using equation (4.1). This is the difference in the utility curve with the query in the pool and with the query removed. Equation (4.7) describes the ongoing value of a query. Where q_i is the query that is being evaluated for removal. $\overline{U}(R,Q,A)$ has also changed slightly with the inclusion of A. A is the set queries that have been made by the system and then abandoned. Although these queries have no-probability of returning a result, we still need to include the monetary cost of make the query and include that in the cost function described in equation (4.3).

$$\Delta_{ongoing}(q_i) = \overline{U}(R,Q,A) - \overline{U}(R,Q-\{q_i\}, A \cup \{q_i\}) \tag{4.7}$$

Evaluating the *replacement value* of the query is also relatively simple using the equations defined in section 4.2 but much more expensive computationally because we need to start considering pairs of queries, the query being added to the pool and the query to be removed from the pool. It is important to note that the query with the highest value (calculated using equation (4.1)) may not be the same as the query with the highest replacement value (calculated using (4.8)). This is because the value of the query being added may depend on the query being removed. Equation (4.8) defines the replacement value for a query given that you are replacing query q_i with q_j.

$$\Delta_{replace}(q_i, q_j) = \overline{U}(R, Q \cup \{q_j\} - \{q_i\}, A \cup \{q_i\}) - \overline{U}(R, Q, A) \tag{4.8}$$

Again, we have to use the new definition for $\overline{U}(R, Q, A)$ to take into account that the monetary cost of querying q_i has already been paid and will not be returned if we remove the query.

4.4.1 Optimizations for finite outstanding queries

Computationally, we are multiplying the processing required by the number of entries in the querying pool. In the case of gathering sessions with a large number of outstanding queries the additional computational cost may have too much of a negative impact on the performance of the system. We have found two methods for reducing this computational overhead without degrading performance too greatly.

The lowest cost method is to calculate the $\Delta_{ongoing}$ for each active query and to remove any queries that fall below a particular threshold, γ. The value γ can be set so that we do not have a situation where the querying resources are exhausted. This method does not take into account the query that is being considered, but is relatively cheap to execute.

The next method is to use the $\Delta_{ongoing}$ calculation to create a subset of queries in the query pool that will be used in determining the $\Delta_{replace}$ value. The size of this subset can be set to an arbitrary n based on the computational speed of the system. We can do the same with the pool of potential queries by selecting a subset based on the highest Δ.

Both of these optimizing approaches do not consider unlikely cases in which removing one query might greatly increase the value of a potential new query, but this situation occurs very rarely. Usually, the value of the query, $\Delta(q_i) = \overline{U}(R, Q \cup \{q_i\}) - \overline{U}(R, Q)$, compared to the value of the query with a low-valued query removed from the query pool, $\Delta(q_i) = \overline{U}(R, Q \cup \{q_i\} - \{q_k\}) - \overline{U}(R, Q - \{q_k\})$, are very close.

CHAPTER 5

COMPONENTS OF A VALUE-DRIVEN SYSTEM

The value-driven information gathering system has four major components that are used to evaluate a set of potential queries and evaluate the expected performance of the current query pool (see **Figure 3.1**). These are the decision model, the value-driven information gathering planner, the information sources database and the information retriever. The system is designed in such a way that each of the four components communicates with each other through a simple set of protocols so that the system can be expanded in the future.

5.1 Decision model

The decision model component manages the decision model described in section 3.1. This includes returning the value of information for an information source, managing multiple instantiations of the decision model, and incorporating evidence when an

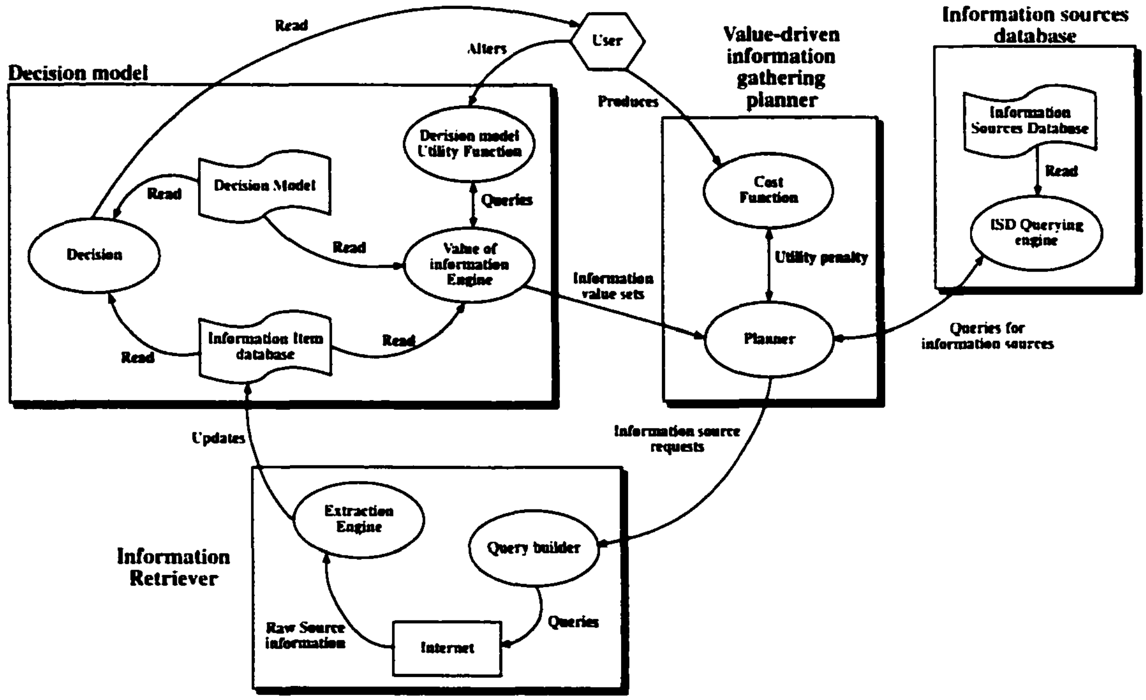


Figure 5.1 - The components of a value-driven information gathering system

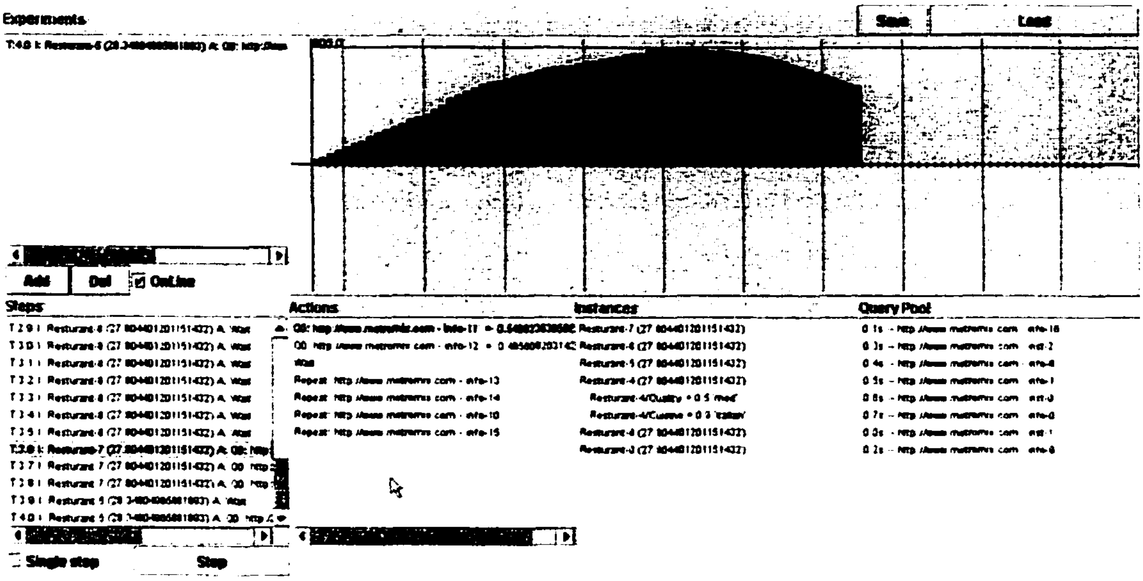
information source returns a value. **Figure 3.4** shows the belief network editor and **Figure 3.10** shows the utility function editor for the restaurant domain.

The expert designing the system would first layout the belief network for each instance using the belief network editor. Next, the expert creates a utility function defining the value of selecting an instance using the belief network.

5.2 Information sources database

The information sources database component manages a database containing interface information, extraction information, access cost, and the response-function for each information source that the value-driven information gathering system may access. **Figure 3.16** shows the tool used to construct the information sources database. Modeling information sources using a response probability at any given time worked very well. While there were sometimes global effects that applied to all of the response-functions (if Internet traffic was high because of the time of day), the general shape of the response-function was consistent over the few weeks that the tests were run on each system. The information source editor allows the expert to add new sources to the value-driven system, measure their responsiveness, specify what information the source returns, set the reliability of the source and attach an extractor to the information source. The information sources editor also allows the expert to specify that a source is a link/instance source, which can be used to generate new instances for the decision model and new information sources for the value-driven system to query.

The extraction editor in the information sources database is for the most part an implementation of Knoblock's automated extraction system (see [4]). This dissertation does not focus on wrapper creation, but we selected Knoblock's system for its ease of use and implementation. See section 6.6 for more details on the exact implementation of extraction system used in the prototype information gathering system.



T 4 0 I: Resturant-5 (28 3-480-4086481803) A: OB: http://www

T 2 9 I: Resturant-8 (27 804401201151432) A: Wait
T 3 0 I: Resturant-8 (27 804401201151432) A: Wait
T 3 1 I: Resturant-8 (27 804401201151432) A: Wait
T 3 2 I: Resturant-8 (27 804401201151432) A: Wait
T 3 3 I: Resturant-8 (27 804401201151432) A: Wait
T 3 4 I: Resturant-8 (27 804401201151432) A: Wait
T 3 5 I: Resturant-8 (27 804401201151432) A: Wait
T 3 6 I: Resturant-7 (27 804401201151432) A: OB: http:
T 3 7 I: Resturant 7 (27 804401201151432) A: OO http
T 3 8 I: Resturant 7 (27 804401201151432) A: OO http
T 3 9 I: Resturant 5 (28 3-480-4086481803) A: Wait
T 4 0 I: Resturant 5 (28 3-480-4086481803) A: OO http

OB: http://www.metromix.com - info-11 = 0.548023838686
OO http://www.metromix.com - info-12 = 0.485608203142
Wait
Repeat: http://www.metromix.com - info-13
Repeat: http://www.metromix.com - info-14
Repeat: http://www.metromix.com - info-10
Repeat: http://www.metromix.com - info-15

Resturant-7 (27 804401201151432)
Resturant-6 (27 804401201151432)
Resturant-5 (27 804401201151432)
Resturant-4 (27 804401201151432)
Resturant-4/Quality = 0.5 'med'
Resturant-4/Cuisine = 0.3 'italian'
Resturant-4 (27 804401201151432)
Resturant-3 (27 804401201151432)

0.1s - http://www.metromix.com - info-16
0.3s - http://www.metromix.com - inst-2
0.4s - http://www.metromix.com - info-4
0.5s - http://www.metromix.com - info-1
0.6s - http://www.metromix.com - inst-0
0.7s - http://www.metromix.com - info-0
0.0s - http://www.metromix.com - inst-1
0.2s - http://www.metromix.com - info-8

Figure 5.2 - The state of the gathering session after an information source and an instance source have returned

5.3 Communication layer

The communication layer maintains the pool of queries that have been sent out but have not yet been answered. It also uses the interface and extraction information provided by the information sources database in order to communicate with the information sources and translate their results to a form that the decision model can use. The information is sent to the decision model as a set of {instance / feature / value / reliability} tokens that are incorporated into the decision model to improve the quality of the decision.

5.4 Value-driven information gathering planner

The value-driven information gathering planner monitors the query pool, the decision model, the information sources database, and the user preferences to decide which, if any, queries to make at any given time. It also decides when to halt the information gathering process and return a decision. The algorithm for making these decisions is described in detail in the previous chapter. **Figure 5.2** shows a trace of the value-driven system during a gathering session.

The trace window in **Figure 5.2** has six components. The first component, in the top left, is the list of **Experiments** recorded by the system. In this window, only one experimental run of the system has been done at this point. The gathering session has returned a decision to select the fifth restaurant for the list of restaurant instances that it retrieved over the course of the gatl.ering session. The second component, in the bottom left of the window, gives a history of the queries and responses that occurred during the gathering session (**Steps**). The highlighted item in the list is a query that was launched at time 3.6 seconds. To the right of the Steps list is the **Actions** list, this list contains a list of all of the potential actions that the system could perform during this time step, sorted by the value of the query for each action. For example, querying the eleventh information source from the MetroMix site is expected to increase the overall expected utility of the gathering session by 0.549 utils, whereas querying the twelfth site listed on the MetroMix site is only expected to increase the overall expected utility of the gathering session by 0.465 utils. To the right of the Actions list is the **Instances** list, which contains a list of all of the restaurant instances that the system is current aware of, as well as any evidence that has been returned related to that instance. At this point, two pieces of evidence have returned relating to the fourth restaurant found during the information gathering session. To the right of the Instances list is the **Query pool** list. This list contains the queries launched by the system that have not yet returned information. In this example, the value-driven system made a query to information source 16 at time 0.1 seconds that has not yet returned. The value-driven system also made a query to instance source number two at time 0.3 seconds that has not yet returned any new instances to evaluate. Finally, above the action, instances, and query pool list is the **Expected utility curve window**. The utility curve window displays the expected utility for the gathering session relative to the current time and the relative to the expected utility of the current best decision the gathering session would make if the gathering session were to halt immediately. You can see that the system expects the utility of the decision made by the system to improve until time 8.2 seconds (Each vertical gray line is a second mark, starting at 4.0 since the system is currently at second 3.6). After 8.2 seconds, the resource cost (time) begins to bring the expected utility curve down from its maximum. As the gathering session proceeds the expected utility curve changes to reflect new queries, new responses and new resource costs.

The trace tool was extremely useful in observing the behavior of the value-driven information gathering system during its development and testing. Section 6.7 shows a trace of a simplified gathering session using the trace window at several points in time.

5.5 Implementation

The original prototype system used for all but the last experimental run (section 6.6) was written in Lisp and implemented on a Sun Ultra. Until the experiment described in section 6.5 we also used HUGIN, a commercially available belief network system. Unfortunately, as we began to add instances and evidence nodes, I needed a system that allowed a more flexible structure for the influence diagrams and had to write my own belief network system.

The final experimental run implement in Java 1.2. Java was chosen because of its cross-platform capability, its extensive library of Internet access functions, and its standard user interface library. The Java implementation was run on a 350 MHz Pentium II system to generate the data used in the restaurant selection system (section 6.6)[10].

[10] It was surprising to see how little speed improvement there was from moving to Java from Lisp.

CHAPTER 6

EXPERIMENTAL RESULTS

This chapter describes the five experimental systems that have been developed to test the value-driven information gathering algorithms describes in Chapter 4. Section 6.1 discusses the testing methodology, which is used for each of the five systems. Section 6.2 through 6.6 describes each experimental system that was built, the aspects of the value-driven information gathering they were designed to test, the problem domain, and the experimental results. Section 6.7 traces the final system making a restaurant selection in a reduced information environment.

6.1 Testing methodology

Two steps were required in order to make statements about the overall effectiveness of value-driven information gathering. First, a set of comparison systems needed to be developed that examined a range of information gathering strategies, which varied in their computational complexity. The second step was to develop a set of metrics to compare the systems.

Section 6.1.1 describes in detail all of the comparison systems that were developed to provide a set of base-line information gathering approaches. Section 6.1.2 describes the metrics that were used during the experiments that were run on the value-driven information gathering system at that stage of development and the comparison systems. Section

6.1.1 Comparison systems

One of the more difficult aspects in evaluating the abilities of a value-driven information gathering system is the fact that no pre-existing information gathering systems exist to serve as a base-line for comparison. My solution was to create a set of several less computationally expensive information gathering systems to use for comparison. For the five experimental systems described below, a variety of comparison systems were created to evaluate a value-driven information gathering system. The goal was to develop a set of comparison systems that represented a spectrum of both performance and computational complexity in deciding which information sources to query. These comparison systems create a context in which the reader can position the value-driven information gathering system in this spectrum and demonstrate its advantages.

1. **Model only**

   The model only approach was used as the most basic of base-line comparison systems. It demonstrates the performance of a system that only uses the decision model itself, with no evidence, to decide on the action with the best expected utility.

2. **Coverage**

   The one approach that many system use is a *Coverage* approach [4][5][14][18][20][21][25][28][53][67]. A coverage approach is extremely straightforward, attempt to gather as much relevant information as possible in the time allowed. Since I would be testing the information gathering systems in environments in which they would purposely *not* have the resources required to query every relevant information source, the coverage system was designed to query information sources in a random order. This was done in order to prevent the intrinsic querying order of the system from having any effect on the comparison.

3. **Feature sorted utility**

   Feature sorted utility used the value of information to sort the information sources from most important to least important offline. Each potential query would have the value of information determined on a decision model with no other evidence having been returned. This gave a base-line value of information to associate with each query. The base-line value of information would then be used to sort the queries in descending order so that the queries with the highest base-line value of information would be queried first. This is an ideal *Coverage* approach because it attempts to query all information sources until the system has exhausted its resources. The ordering of the queries is optimal, given that the system can only determine the value of information for each query in an off-line setting. A feature sorted utility system will begin by querying the information sources with the isolated highest value of information and ending with the query, which has the lowest expected impact on the quality of the decision.

4. **VOI only**

   Only the value of information was used to determine which information source to query next. This approach did not take into account the expected resource cost of querying an information source only the expected improvement in decision quality once the information source returned. If we were in an environment in which information queries were returned immediately and had no fixed querying cost (for example, a monetary cost to query the site) this would be equivalent to value-driven information gathering. This comparison system varies from the *feature sorted utility* system in that the value of information for an information source can vary based on other information that has been returned in this system. The *feature sorted utility* uses the value of information for an information source with an empty decision model and continues to use those values that were calculated prior to the information gathering session.

5. **"Ideal" system**

   An ideal system is defined as a system that immediately knows the result of every possible query instantly and at no cost. It would always make the correct decision (assuming that the information sources contained accurate information) using no resources. The ideal system provides us with an upper bound on performance for any information gathering system that makes a decision. Such a system could not actually exist, but it is a useful comparison system to use in order to see how close value-driven information gathering (as well as the other comparison systems) is to a perfect information gathering and decision system.

Many of the experiments below use a subset of these comparison systems. This is primarily because the experiments described below took place over several years of research and the specific base-line comparison systems changed as the research received feedback. The selection of comparison systems also changed based on which aspects of the value-driven information gathering system were being tested at the time.

6.1.2 Evaluation metrics

For each of the following five experimental systems, I have chosen to use two metrics to compare different information gathering strategies: The average information utility and the accuracy compared to a system with perfect information.

The *average information utility* metric uses the influence diagram itself and the evidence that has been returned by each strategy in order to calculate the expected utility of the maximizing decision when the system decides to halt information gathering and return a decision.

The *accuracy* metric is to compare the average percentage chance that the specific strategy being evaluated has made the same decision as a system with perfect information (the "Ideal" system described above).

The advantage of using the average information utility, compared to only using the accuracy percentage, is that it takes into account the potential cost of making a near-accurate decision. The utility function for a decision model can express situations in which selecting the second best object to purchase is very close in utility to selecting the best object. An accuracy rating cannot express this aspect of an environment.

6.1.3 The Simulated Internet Environment

The experimental systems described in sections 6.2 and 6.3 both operated in a simulated internet environment. E.rly in the work on value-driven information gathering, the tools did not exist to extract and incorporate information from the web into the decision model. Instead, an Internet simulator was created that had a number of information sources with unique response-functions and evidence directly attached to the information source. When a simulated information source was queried, the time at which the information source would respond was randomly determined using the response function. When the information sources responded, it would return a list of feature / state pairs that were used to instantiate variable nodes in the decision model.

The response-functions for these simulated information sources were randomly generated using statistics gathering from actual web sites on the Internet. Early in the creation of the simulated internet environment, a program was written that repeatedly queried a set of web pages and gathered statistics about when the page would return. The web page polling program would store these statistics as a histogram of response times for a specific site (see **Figure 6.8**).

Although the simulation would randomly generate response-functions for an information source, these response-functions had distributions that closely matched the distributions of many of the web site that were polled. In general, the response-function for a web site is a large Gaussian curve followed by a number of smaller Gaussian curves (see **Figure 6.8**). Although the web page polling program would not show the path traveled by the response packet, I believe that these different Gaussian curves correspond to different routers that the query and the response went through over the course of a round-trip from the value-driven system to the server being sent the query. The web page polling tool makes me very confident that the experimental data collected from the two systems that used the simulated internet environment are as valid as the results from the three other systems, which used the internet to gathering information.

6.2 Sequential querying

The first value-driven information gathering system that was developed initially only queried one information source at a time. It assumed that the information source returned perfect information, so the value was used to directly instantiate the decision model. This first system was tested in a simulated environment in which the response times of web pages were modeled from samples we had taken from web pages. This system maintained the ongoing value of the query for the one query that was active and compared it to the other potential queries to see when it was time to abandon the current query and make a new one if the current query had not returned a value. This initial value-driven system made one of four choices every time through the querying loop:

- **Begin querying a new site:** At the start of the querying session, or when a query has returned, the value-driven system chooses a new query to launch based on the value of the query.
- **Abandon the current query and make a new one:** When the ongoing value of the query is less than the replacement value of a new query, the current query is halted and a new query is made. Generally, this occurs for one of two reasons: First, because a query has not returned by a point in time, the odds of it returning in the future are very slim, so the expected benefit to the decision decreases to a point where it is worthwhile to abandon the query. Second, information from another source has returned that makes the information that will be returned by the query irrelevant. Although, the average amount of time for a query to return and be extracted is small (usually one to ten seconds) these two events do occur and a value-driven system can conserve querying resources by abandoning these non-productive queries.
- **Continue querying the current site:** When the expected utility of the current query is greater then the expected utility of launching any new queries, the system waits for the current query to return a value.
- **Stop execution and return a result:** When the expected utility of the current query or any potential queries is negative, then the system returns the current decision.

Minimum time	**Deadline time**	**Random selection utility**	**Feature sorted utility**	**VDIG utility**	**"Ideal" system**
Set A					
1.0 sec	2.0 sec	0.830	1.283	1.931	4.2
2.0 sec	5.0 sec	1.390	2.341	2.750	4.2
Set B					
4.0 sec	8.0 sec	1.858	2.432	2.941	4.2
5.0 sec	10.0 sec	3.189	3.735	4.076	4.2

Table 6.1 - Results of the car purchasing experiments

6.2.1 Domain: Car purchasing

The problem domain for this system was a purchasing decision about a car. The system did not have specific instances, so this system only returned the expected utility of purchasing the car. The decision model contained six features:

- The manufacturer of the car
- The retail cost
- The gas mileage
- Four wheel drive
- Depreciation
- Anti-lock brakes

The system simulated 25 information sources that returned one or more pieces of information to the system. Each information source had a randomly generated response-function and a monetary cost for querying the site. Each time the experiment was run the values for each feature of the car would vary.

6.2.2 Evaluation

The value-driven information gathering system was tested against two simpler query strategies: A random querying algorithm and a querying algorithm sorted by the feature value. In each experiment, the system ran 500 times and the results were averaged.

The values in each column represent the average information utility generated by executing the value-driven system, a random querying selection system, and feature-sorted systems. In all of these examples, the VDIG strategy generated better information for the user in the same amount of time. As the time was extended the difference becomes less noticeable because eventually all three systems had enough time to query most or all of the information sources. The maximum score that a system could achieve was 4.2. As the resources given to the system increases (in this case the amount of time the system is allowed to use) the difference between each of the three strategies decreases (see the last row of **Table 6.1**). At this point all three systems are able to query a majority of the information sources before the deadline time. Thus, the quality of the decision made by all three systems approached the maximum of 4.2 as we increased the amount of time each system was allowed to use before returning a decision. Although a ten second maximum for an entire information gathering session may seem extremely short, it is worth noting that these systems were still operating in a simulated information environment in which no post-processing of information sources needed to occur. Because of this, once an information source returned a value it could be instantly incorporated into the decision model.

This system demonstrated that a value-driven approach could improve information gathering performance in a very simple domain. The next step was to expand the information gathering mechanism to be able to query multiple information sources simultaneously in order to better approximate what an information gathering system that operated on the internet could do in the real world.

6.3 Parallel querying

The next system created to test value-driven information gathering could query multiple information sources in parallel. It still worked in a simulated environment and it still assumed that the information returned by an information source was perfect. This experimental test-bed also did not implement instances, so it only evaluated one decision and returned a recommendation. The influence diagram used in this set of experiments contained several more variable nodes than the influence diagram used in section 6.2 and the information sources were more accurately modeled. See section 6.1.3 for a description of how the response-functions for the information sources were being modeled. The response-functions for the information sources were more accurately modeled by the inclusion of different response-function for different times of the day. These divisions were rough (only morning, afternoon and evening), but I wanted to model the effect that time-of-day had on the response-functions. The web page polling tool described in section 6.1.3 was modified to include time of day information and construct separate response histograms for each time-period. These histograms were then used to create different distributions for the simulated information sources for each time period.

6.3.1 Domain: Job evaluation

The second value-driven system evaluated a decision on taking a job that has been offered. The job decision gathered information from the Internet on a number of factors that influence the decision on whether to take a job or not. **Figure 6.1** shows the influence diagram used in making the job decision. The system evaluated the benefits from the job itself, the area the person would have to relocate, and the housing costs. The job decision was also based on the overall housing quality, the desirability of the job, the disposable income after taking the cost of living into account, the safety of the area and the quality of education in the area. Although the system was still working in a simulated Internet environment, we were able to find all of the above information on the Internet. **Figure 6.2** shows the job comparison running and **Figure 6.3** shows the response-function for one of the information sources.

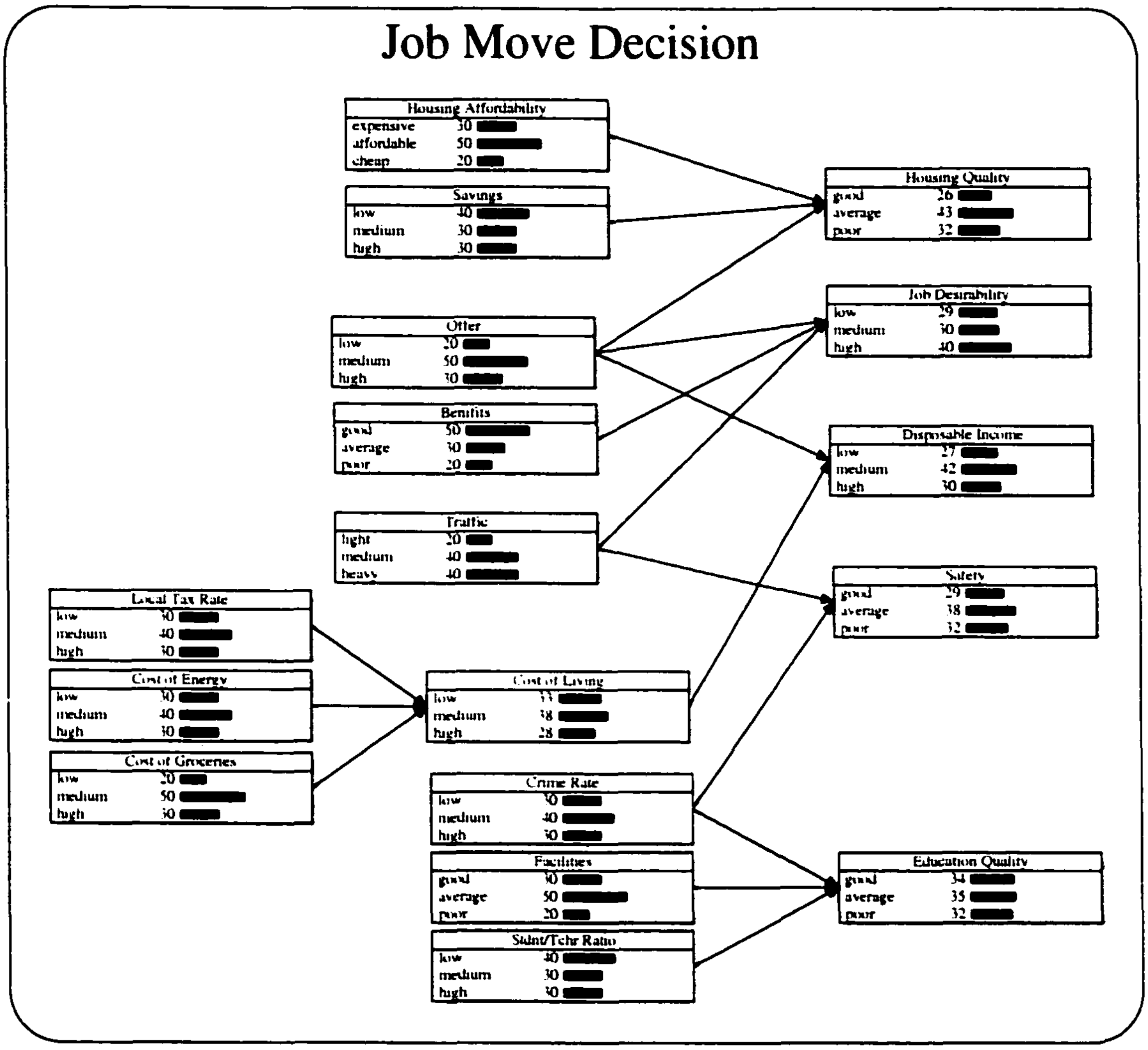
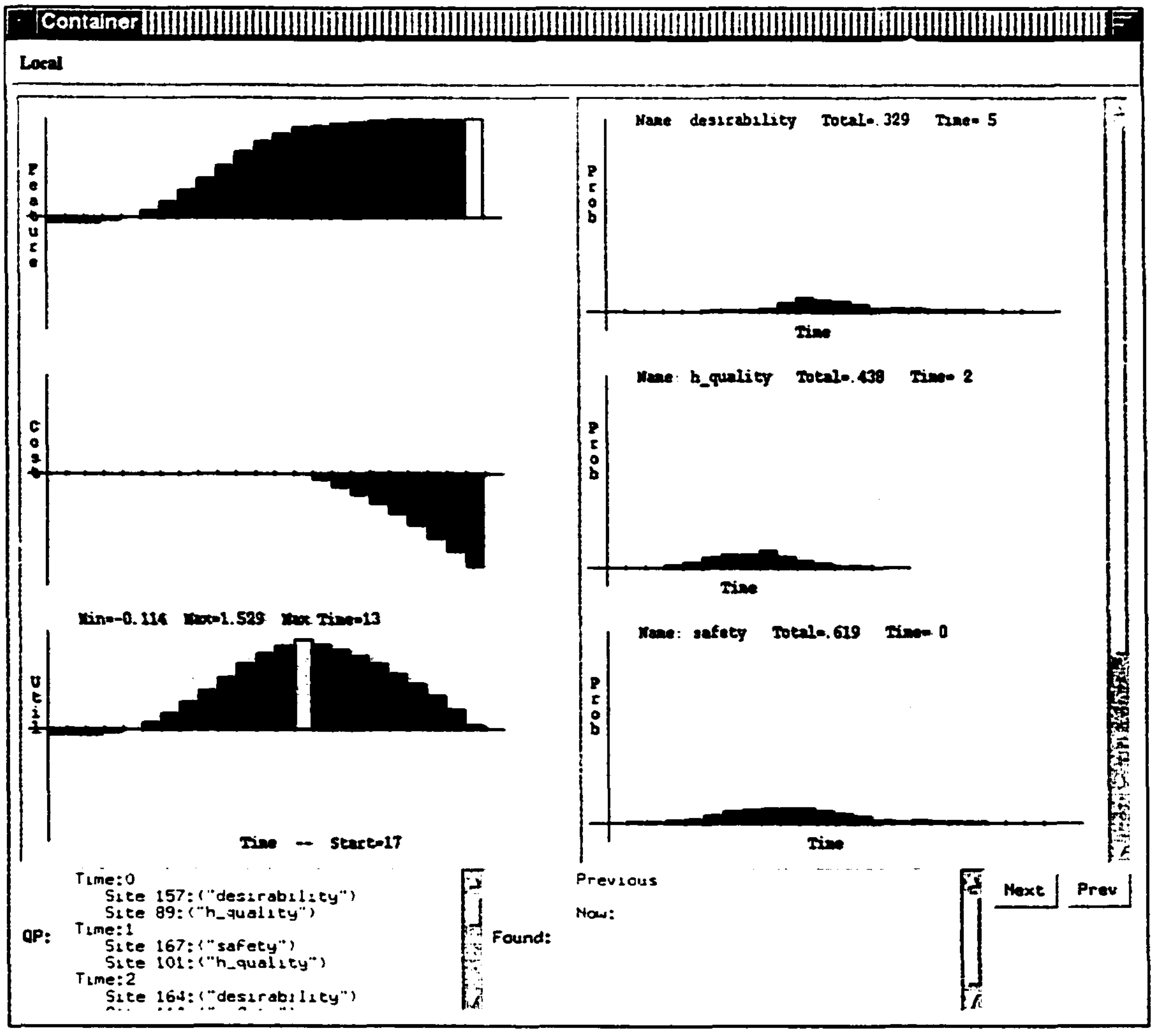


Figure 6.1 - Job decision influence diagram

While no one is going to trust a piece of software to make a decision as important as whether to take a job or not, the job move decision could easily be used for three important tasks that people looking for a new job need. The first task is to act as a filter; the job move decision system described here could automatically process a large selection of jobs and return those with an expected utility greater then some threshold. The second task is to recommend jobs and show the user the evidence supporting that expected utility. The user could use these web sites as a starting point for making a more thorough evaluation on their own. The third task, which this system could be used for, is for comparing sets of jobs and displaying the differentiating evidence to the user for further study. For example, the system could inform the users that while both jobs have the same benefits, one has better schools, and the other has lower housing costs.

6.3.2 Evaluation of Job Move decision

The system was compared against three other information gathering strategies (see **Table 6.2**). The first information gathering strategy, *Model only* (Column one), is the decision that the influence diagram would make with no information. In this case, that decision was to decide to make the move for the new job. Column two is the score for


Container
Local
Measure
Cost
Min=-0.114 Max=1.529 Max Time=13
Util
Time -- Start=17
Name desirability Total=.329 Time= 5
Prob
Time
Name h_quality Total=.438 Time= 2
Prob
Time
Name: safety Total=.619 Time= 0
Prob
Time
Previous
Now:
Found:
Next
Prev

```
QP: Time:0
       Site 157:("desirability")
       Site 89:("h_quality")
    Time:1
       Site 167:("safety")
       Site 101:("h_quality")
    Time:2
       Site 164:("desirability")
```

Figure 6.2 - The value-driven information gathering system running for the job evaluation experiment

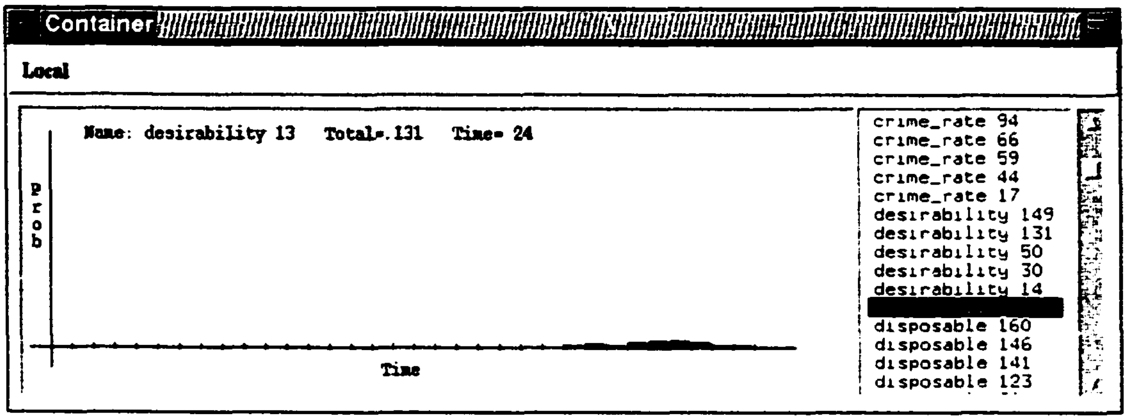


Figure 6.3 - A response-function for an information source in the job evaluation experiment

an information gathering strategy called *Coverage*. Coverage attempts to find information about each variable node in the influence diagram without taking into account the value of information or the response-function. A coverage approach is less computationally intensive than value-driven information gathering, but the difference in execution time is less than 1% of the total execution time for the system. The fourth column in **Table 6.2** , *"Ideal" system* represents an ideal system where all the information used in the decision model is known immediately. The experiment was run 500 times and the values for the variable nodes in the job influence diagram were randomly selected at the start of each experiment. The utility score represents the average utility of the decision.

6.3.3 Domain: Software purchasing decision

We implemented another experiment with the same system for making software purchasing decisions. **Figure 6.4** shows the decision model used in deciding whether to

	Model only	Coverage	VDIG system	"Ideal" system
Utility	0.900	1.150	2.066	3.09
Accuracy	59%	60%	86%	100%

Table 6.2 - A comparison of different information gathering strategies for the job evaluation experiments

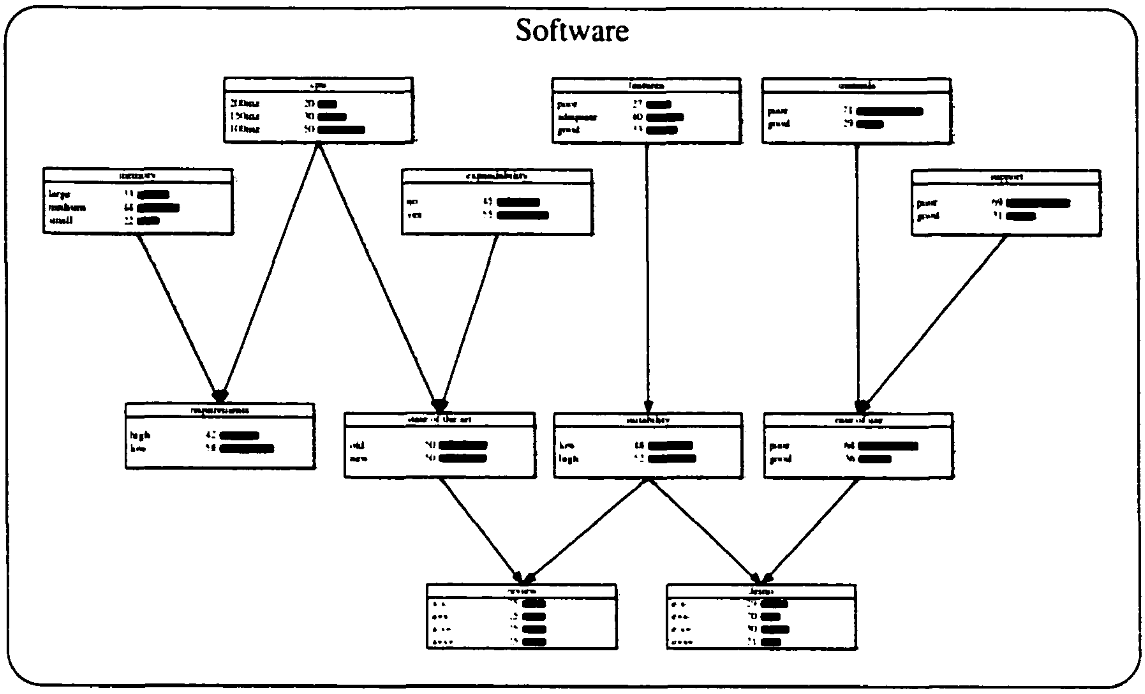


Figure 6.4 - Software purchasing influence diagram

purchase a piece of software or not. Information that was used for evaluating the software included the requirements of the software, its suitability, ease of use and cost. Although this experiment was done in the simulated Internet environment, we were able to find all of this information at various sites on the Internet.

6.3.4 Evaluation of Software purchasing decision

We compared the system to the *Coverage* approach as before for directing information gathering as well as the score for *Model only* (The base decision given no information was to purchase the software) and the score for perfect information (*Ideal system)*. The experiment was run 500 times and the results were averaged, with the true

	Model only	Coverage	VDIG System	"Ideal" system
Utility	1.120	2.090	3.055	4.040
Accuracy	60%	62%	76%	100%

Table 6.3 - A comparison of different information gathering strategies for the software purchasing experiments

values for the software package randomized at the start of each run. Value-driven information gathering scored significantly above the coverage approach while spending the same resources. **Table 6.3** shows the performance of the value-driven information gathering system (VDIG) compared to three other approaches (see section 6.1.1 for more information on the comparison systems). Performance was measure in two ways. The first measure was the average utility of the software product that was selected (the **Utility** row). The second measure was how often the system selected the system with the highest utility (the **Accuracy** row). In the case of the *Model only* and the *"Ideal" system*, the systems returned a result immediately. The *Coverage* system ran until the cost of time was non-zero (in this case 30 seconds) and the *VDIG System* ran until the system decided to halt and return a value, paying a resource penalty for any time used beyond 30 seconds. In these experiments, the system would usually return a result without using any additional time because the cost of using additional time did not outweigh the expected increase in decision quality.

6.4 Implementation on the Internet

The next step was to build a text-extraction engine and implement the system on the Internet. We found that the time required for extracting information from the web pages was not trivial. In general, the time required to run an extraction wrapper on an information source was around 10% of the time required to query and receive the information from the source. To take the extraction time into account, the system

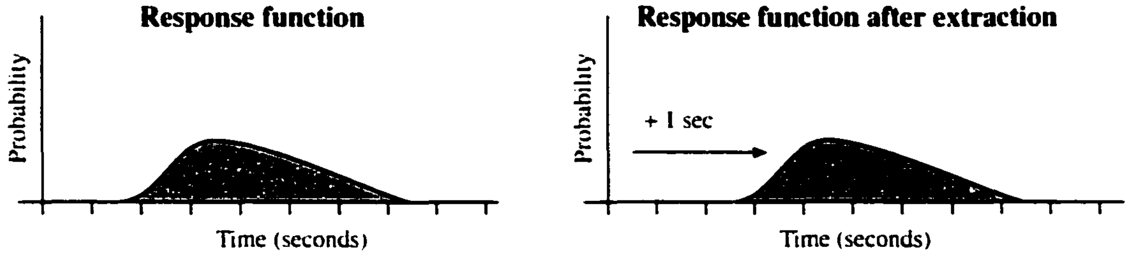


Figure 6.5 - Expected utility including the extraction time for the digital camera experiment

incorporated the extraction time into the response-function for the information source. For example, if extracting a specific web page took one second, the response-function would be equal to the response for querying the web site shifted by one second (see **Figure 6.5**).

6.4.1 Domain: Digital camera selection

For these experiments, we built a decision model for purchasing a digital camera. **Figure 6.6** shows the decision model. We have constructed a simple decision model representing the usefulness of the key features of a digital camera. For example, if the camera uses flash storage cards than the system has greater storage capacity. The low-

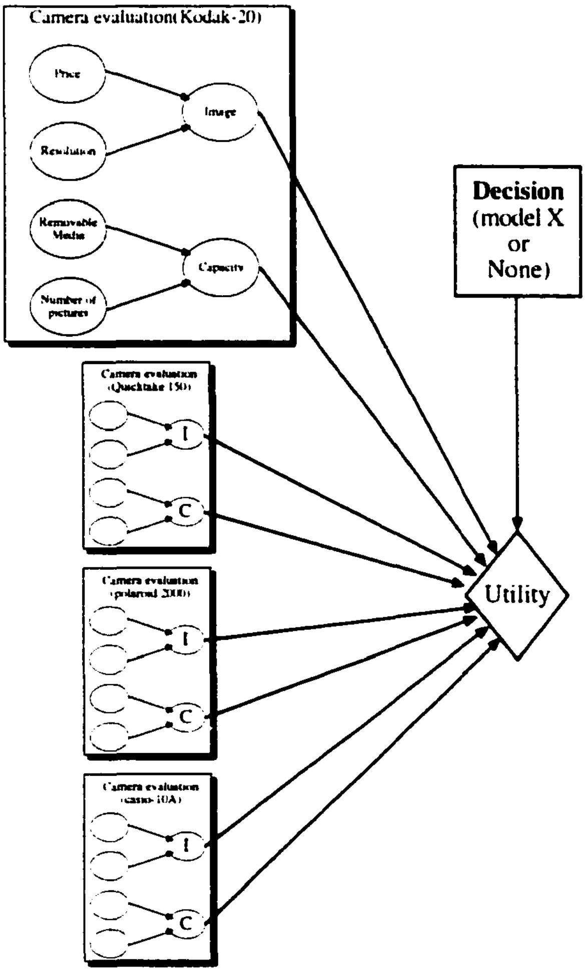
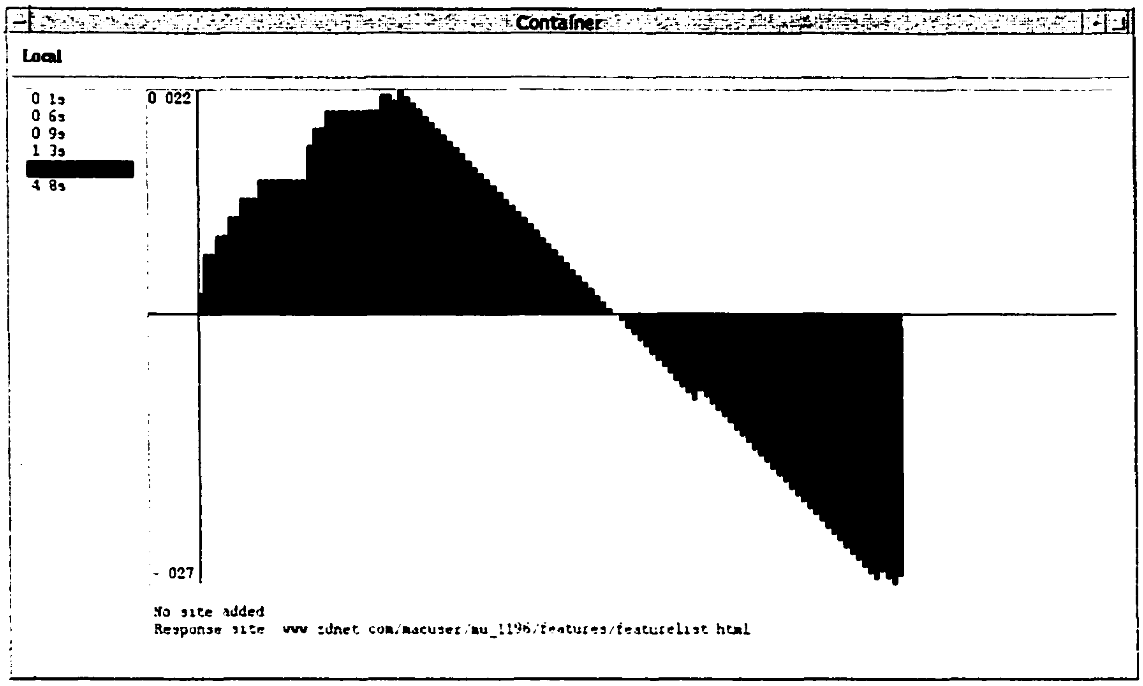

Camera evaluation(Kodak-20)
Price
Image
Resolution
Removable Media
Capacity
Number of pictures
Decision (model X or None)
Camera evaluation (Quicktake 150)
I
C
Camera evaluation (polaroid 2000)
I
C
Camera evaluation (casio-10A)
I
C
Utility

Figure 6.6 - Digital camera decision model



No site added
Response site www zdnet com/macuser/mu_1196/features/featurelist html

Figure 6.7 – The VDIG working on a camera decision

level features are then used to evaluate high-level features about the camera. The relationship between low-level features and high-level features is defined by an expert. In this example, this was done in order to make specifying a utility function easier for a novice user (e.g. I want a camera that is expandable, but is a camera that uses 8MB flash cards more expandable then one that uses floppy disks?). In other domains, it may be possible to gather some high-level information directly, giving the system two options: Attempt to directly retrieve the high-level information, or attempt to extrapolate it from low-level features.

In our experiments, the system had five choices, to choose one of the four cameras presented to it, or to decide not to purchase any of them. In each experiment, the system had ten information sources available to query. **Figure 6.7** shows the output of one of the test runs. The graph is the future expected utility cure for the gathering session. At time 1.9 seconds, the site www.zdnet.com/macuser/mu_1196/features/featurelist.html has returned a value.

Figure 6.8 shows the response-function for one of the information sources used in the camera decision. The value-driven system uses the response-function of the

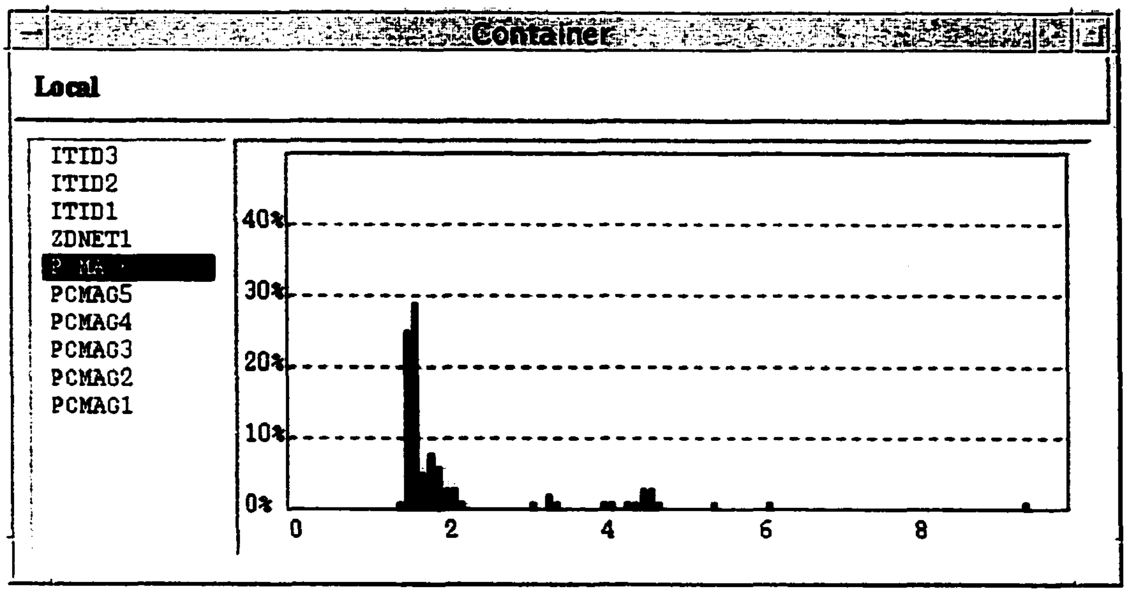


Figure 6.8 - The response-function for an information source in the camera experiments

information source as well as the value of the evidence returned by the information source to calculate the expected benefit of making a query to the information source. The precise role of the response-function is described in equation (4.6). The graph in **Figure 6.8** shows the values of $\Pr_r(i)$ where the height of each bar is the probability of the information source returning and the horizontal position of each bar is the time after the query has been sent, measured in seconds.

We found that the simulated Internet environment was a very good match for running the system on the real Internet (see section 6.1.3 for more information on the simulated internet environment). Switching between simulated and non-simulated mode in the communication layer had no effect on the results that we obtained.

6.4.2 Evaluation

The value-driven system was compared against three baseline systems: The first system attempted to collect information for each item in the decision model with equal weight (coverage). The second system used the value of information, but did not use any information about the information sources in order to decide which sources to query

	Coverage	VOI only	VDIG	"Ideal" system
Utility	0.122	0.213	0.255	0.295
Accuracy	46%	58%	80%	100%
15 seconds of time for free and a utility penalty of 0.1 for every additional second.				

	Coverage	VOI only	VDIG	"Ideal" system
Utility	0.130	0.213	0.251	0.295
Accuracy	44%	58%	80%	100%
5 seconds of time for free and a utility penalty of 0.1 for every additional second				

Table 6.4 - Comparison of VDIG to other retrieval methods for the digital camera experiments

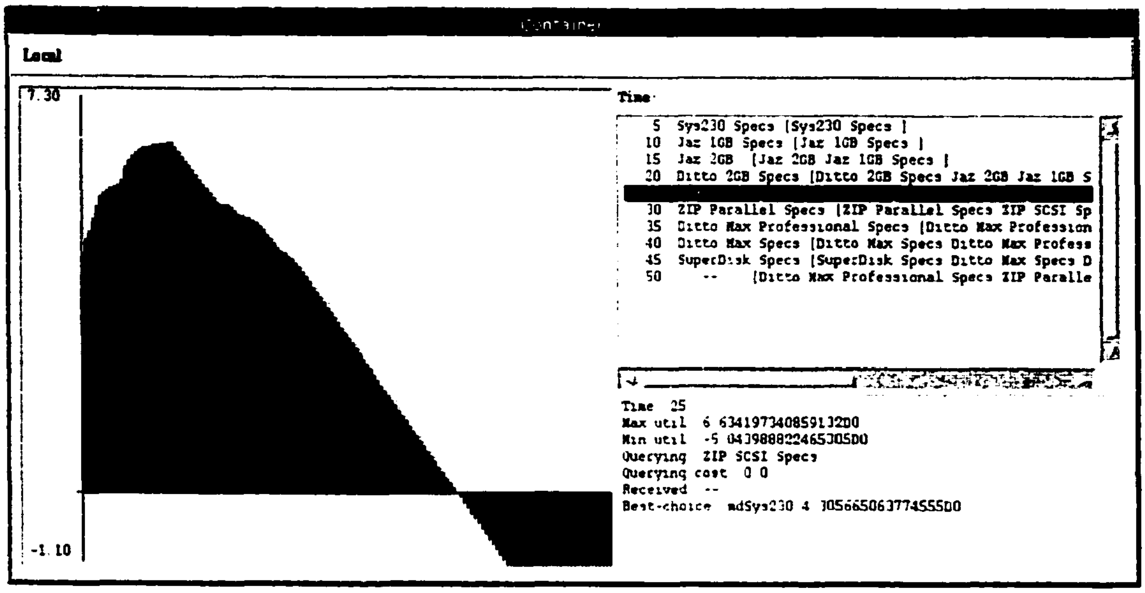
(VOI only). Finally, the third system always returned the correct decision instantly (an "ideal" system). Of course, this third system cannot actually exist, but it is useful to comparison for the other systems. Implementing the ideal system was done by giving the coverage system an unlimited amount of free time and monetary resources and running the system until it had queried all of the available information sources. The coverage and VOI only systems would continue to query until the cost function was non-zero. We tested all four systems under a variety of resource constraints, utility functions, cost functions and available choices of cameras (in total we had ten cameras, but for each experiment we selected four cameras for the system to select from). For each set of resource constraints, (see **Table 6.4**) we ran the system fifty times, each time with a different utility function and set of cameras to pick.

Value-driven information gathering performed significantly better then the other two approaches (coverage and VOI only) while using the same resources. Performance was measured in two ways. The first measurement was the accuracy of the decision, how often did the value-driven system choose the camera with the highest utility. The second measure was the quality of the decision, what is the average utility of the camera selected by the system. Implementing a value-driven system on the Internet was not difficult in our framework; the only change was that a new version of the communication layer actually sent out queries on the internet instead of to our simulator.

6.5 Adding evidence and instances

After the digital camera experiment, it became apparent that the value-driven system was going to need a method for representing conflicting data and a large number of instances. This requirement necessitated a major change in the code base for the project because I could no longer use HUGIN as our belief network evaluation engine. The experimental systems described in section 6.2, 6.3 and 6.4 had all used HUGIN to evaluate the belief networks used in the decision models. HUGIN is a commercially available belief network evaluation engine, which has an integrated set of belief network creation tools and a lisp-accessible library of evaluation functions, which can be applied to the belief networks created by HUGIN. **Figure 6.4** shows a belief network created using the HUGIN belief network editor. HUGIN has the advantage of being extremely fast at evaluating belief networks, but in order to do so must spend some time "pre-evaluating" or "compiling" the belief network once it is created. Because of this, when changes were made to the belief network (for example, adding an evidence node) a costly pre-evaluation function had to be executed in order for HUGIN to be able to evaluate the altered belief network. HUGIN was also unable to handle multiple belief networks with no causal connections. Unfortunately, each instance in the decision model is exactly that, a belief network with no causal connections to the other belief networks. I had found a technique to get around this second restriction for the experimental system described in section 6.4, but it grew increasingly cumbersome as the number of instances grew.

The creation of my own custom belief network evaluation engine (BNEE) was extremely time-consuming, but allowed me to take advantage of many of the structural regularities that I had used in the creation of the belief networks used in our decision models. For example, the BNEE could partially evaluate only the belief networks in the decision model, which had changed at any given instance. The BNEE could pre-calculate the probability distributions for an uninstantiated belief network when a new instance came into existence and use that instead of evaluating an empty belief network. We could increase the speed with which the conditional probability table for the evidence nodes could be generated and propagated through the network because they



```
Time·
    5  Sys230 Specs [Sys230 Specs ]
   10  Jaz 1GB Specs [Jaz 1GB Specs ]
   15  Jaz 2GB  [Jaz 2GB Jaz 1GB Specs ]
   20  Ditto 2GB Specs [Ditto 2GB Specs Jaz 2GB Jaz 1GB S

   30  ZIP Parallel Specs [ZIP Parallel Specs ZIP SCSI Sp
   35  Ditto Max Professional Specs [Ditto Max Profession
   40  Ditto Max Specs [Ditto Max Specs Ditto Max Profess
   45  SuperDisk Specs [SuperDisk Specs Ditto Max Specs D
   50     --     [Ditto Max Professional Specs ZIP Paralle

Time  25
Max util  6 6341973408591J200
Min util  -5 04J988822465J0500
Querying  ZIP SCSI Specs
Querying cost  0 0
Received  --
Best-choice  mdSys230 4 J05665063774555D0
```

Figure 6.9 - The expected utility curve for a run of the removable media experimental system

were always in an instantiated state and we did not have to use the HUGIN lisp library for creating the conditional probability table of the new node. We could integrate our editor into the value-driven information gathering system so that the state of each variable and evidence node for each instance could be tracked and examined during the evaluation process. We also implemented Horvitz's partial evaluation algorithm for evaluating the belief network [41], so it would be possible to make the decision model evaluation an anytime process. Finally, we could maintain a cache of common instance states, which we could use instead of evaluating the belief network. In the end, having our own custom belief network evaluation engine contributed to the speed of the system and my ability to trace through the systems behavior during execution.

6.5.1 Domain: Removable media purchase

For this experiment, we created an influence diagram for recommending a removable-media device. A removable-media device includes items such as Zip Drives, CD-ROM recorders and tape drives. The system queries a subset of 19 information sources and evaluates products based on 26 features and 19 user variables. After

Features	Products	Web pages
Portability	ADrive	Ditto 2GB Specs
AverageSeekTime	Jazz1GB	EZFlyer Specs
Weight	MicroApex	MCA2600 Specs
Warranty	Shark250	Shark 250 Specs
Price	SuperDisk	Sys230 Specs
Capacity	Sys230	UHC3130 Specs
⋮	⋮	⋮
MediaCost	ZipSCSI	ZIP SCSI Specs

Table 6.5 - The products, features, and web sites used in the removable-media evaluation process

querying, the system returns a recommendation of one of 17 products. **Table 6.5** shows some of the features, products, and web pages used by the system. This system and the experimental results are presented in *A Value-Driven System for Autonomous Information Gathering* [34].

The decision model and the utility function simulate the user's preferences for a specific class of removable-media device. For example, some users are willing to sacrifice up-front cost for a lower media cost, while others want devices that can rewrite to media. While the decision model would not change, the user preferences would affect the utility function used to evaluate each product instance. **Figure 6.9** shows the value-driven system in operation during an information gathering session. This image shows a graph of the expected future utility, the active queries, and which product it would recommend if it had to halt immediately.

6.5.2 Evaluation

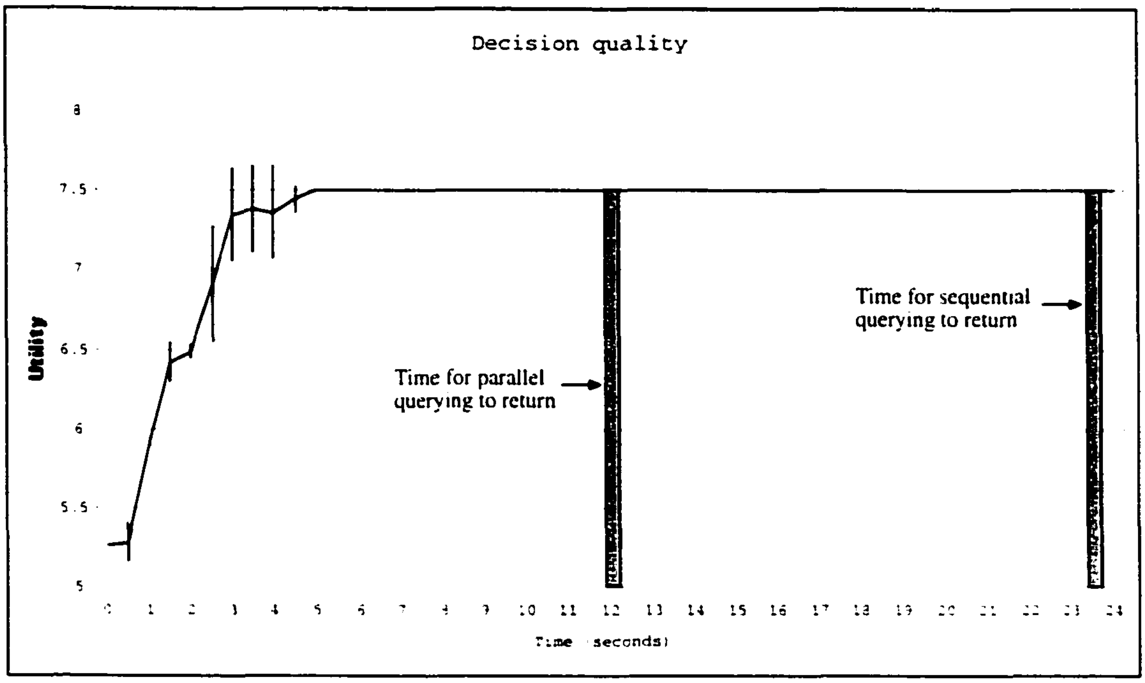
We ran the system continuously over the course of several days and compared the results to two base-line systems. One base-line system queries the relevant information sources sequentially. This is similar to a very efficient human browser who quickly extracts the needed information and moves on. The other base-line system does the same thing in parallel (there is no limit to the number of pending queries, but queries can only be added at a rate of one per half second, just like the value-driven system).

	Value-driven information gathering	Parallel information gathering	Sequential information gathering
Query time (sec)	3.5	11.8	23.7
Expected utility	7.443	7.506	7.506
Accuracy of decision	94%	100%	100%

Table 6.6 - Results of value-driven information gathering

Table 6.6 summarizes the results we got with the three systems. For each system, we measured the average time taken to reach a decision (over 500 runs), the value of the decision and the percentage of time the system made the optimal decision. The optimal choice is defined as the choice the system would make with all available information. Therefore, it is not surprising that all versions converge eventually to the correct decision. However, it is encouraging to see that the value-driven approach reach 94% accuracy within 3.5 seconds compared with 11.8 and 23.7, required by the parallel and sequential base-line system, respectively. Keep in mind that even though the system makes few near optimal decisions the expected utility of these decisions is extremely close to the optimal one.

Figure 6.10 shows the expected quality of the value-driven system as a function of time (its performance profile as an anytime algorithm). The vertical bars represent standard deviation in utility score of the system. The graph shows that the value-driven system quickly improves the decision quality and reaches a high-quality decision within a fraction of the time required by the two base-line systems (marked by the dark columns). The value-driven information gathering system accomplishes this in two ways. First, the system never queries information sources that have such a long expected response time that the resources costs (in terms of time used) exceed the expected increase in decision quality. This is primarily why a value-driven system can return a high-quality decision in a much shorter amount of time then querying the entire set of potential information sources, even if those queries could all be made simultaneously. Second, in an environment in which the information contained by the set of all potential information sources is highly redundant, the value-driven information



Time for parallel querying to return

Time for sequential querying to return

Figure 6.10 - Decision quality over time for the removable media experiments

gathering system can reach a high-quality decision by querying the information sources with the most relevant evidence to the decision. The other two information gathering strategies do not take into account the influence that an information source will have on the quality of the decision. These two features of a value-driven system allows it to restrict the set of queries to those that will return information in the time required, as well as restrict the set of queries to those that will have an substantial impact on the final decision. Restricting the set of potential queries in these two ways greatly reduces the total resources used during the gathering session while still returning a high-quality decision.

6.6 Dynamically altering the decision model

The final experimental system was designed to evaluate the utility of querying *instance sources* as well as implement the previous value-driven work in a new domain. Although the quantitative results are not that different from the previous experiments, the behavior of the system is much more robust. Previous systems needed to have a complete list of all of the information sources available to them and extraction methods for each information source. This restriction was a result of previous extractors being so inflexible. This final system implements a much more flexible system to defining and accessing information sources. The new extraction engine is based on work by Knoblock [4] on semi-automatic wrapper generation. The new extraction engine, which was implemented is more flexible in methods used to extract data from a web page. A description of where to find the text used for evidence can be referenced by a list of previous and next text markers. For example, the text referring to the cost of a restaurant could be referenced by:

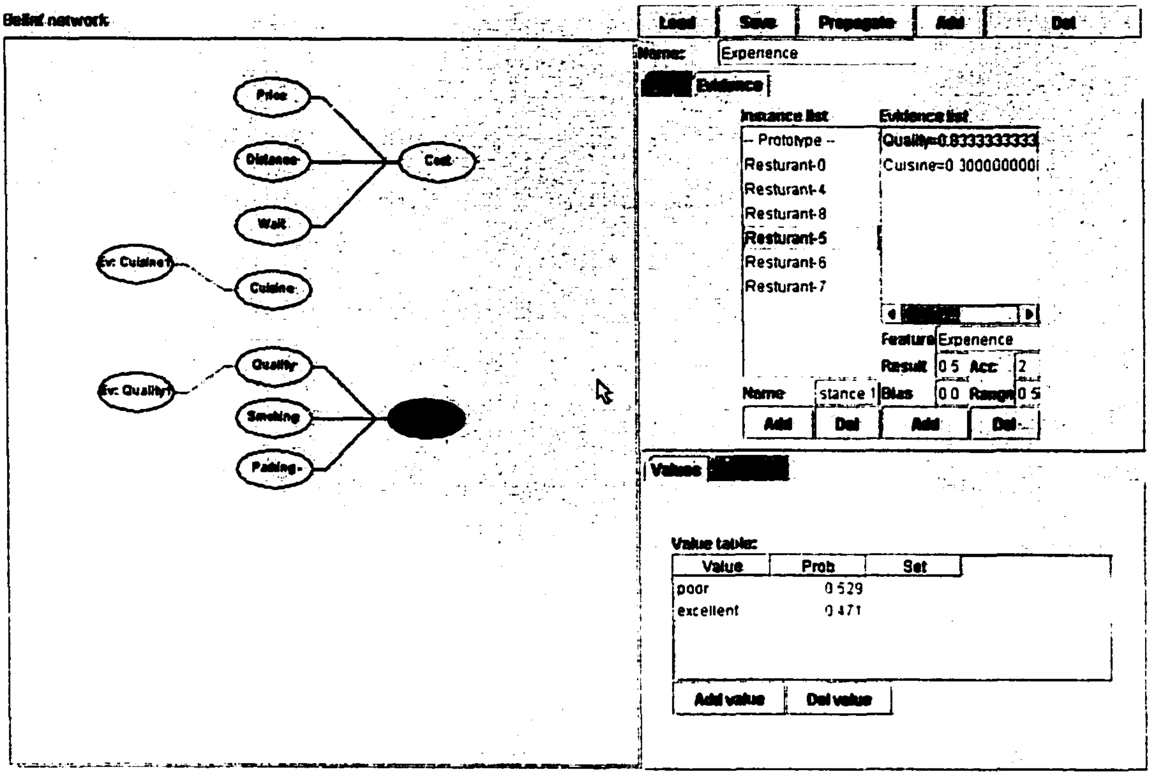
1. Look for the second <H1> html tag
2. Find the matching </H1> html tag
3. Look for the second "Price:"
4. Look for the first <B> html tag
5. Begin grabbing text
6. Until </B> is found

This allows the system to extract text in documents that are much more varied in their contents then the previous extraction engines that we used.

Figure 6.11 shows the decision model for one instance in the decision model. The left half of the window shows the belief network for the instance with the evidence nodes (*ev:quality* and *ev:cuisine*) that have been returned at this point in the gathering session. The table in the top right portion contains a list of all of the active instances and a list of all of the evidence that has been returned for the selected instance. The table in the bottom right portion of the screen displays the probability distribution for the selected node (*experience*) at this point in the gathering session.

The main change in the system is the addition of two new types of information sources: link sources and instance sources.

An instance source is a resource on the Internet that the VDIG system may query to learn of one or more instances that can be added to the decision model. For example, an instance source in the context of the restaurant selection system is a web page that



Name: Experience

Instance list:
- -- Prototype --
- Resturant-0
- Resturant-4
- Resturant-8
- Resturant-5
- Resturant-6
- Resturant-7

Evidence list:
- Quality=0.8333333333
- Cuisine=0 3000000001

Feature: Experience
Result: 0 5 Acc: 2
Name: stance 1 Bias: 0 0 Range: 0 5

Value table:

Value	Prob	Set
poor	0 529	
excellent	0 471	

Figure 6.11 - Decision model for each instance of a restaurant

contains a list of new restaurants that the system may evaluate. In the case of the MetroMix site, when the value-driven system accesses the restaurant guide, there is a list of over fifty pages of restaurants. Each one of these pages linked to by the main restaurant guide lists ten restaurants. Each one of these pages represents an instance source. When a page of restaurants from the restaurant guide is queried, the value-driven system becomes aware of ten new restaurant instances that can be added to the decision model.

A link source is a piece of information that can be used to create a new information source, extractor, and response graph. In the restaurant decision support system the name of the instance and the URL addresses for the information sources where always easy to associate with each other. On the MetroMix site, the name of a restaurant would be part of the link to a web page with its review. For each prototypical review page, we would construct an extractor and measure the responsiveness of the page. These would be inherited by any information sources generated from the page.

The value-driven information gathering algorithm was expanded to allow the system to calculate the two additional values for actions, which the new system could perform. The first value to be calculated was the value of adding a new information source to the set of potential information sources. This is not the same as querying the information source, this value merely represents the expected increase in decision quality by having this new information source available to be queried. The system would still have to pay the resources costs associated with actually querying the site if the system chooses to query it. The second value is the value of adding a new instance to the set of instances, which the value-driven system will evaluate. This value is calculated using the probability that a new instance will have the highest expected utility at the end of the information gathering session. As one might expect, these values are generally much lower then the value of querying an information source. There are three primary situations in which the value-driven information gathering system may decide to query a link source (which returns a set of information sources that may be queried) or an instance source (which returns a new set of instances to be added to the decision model). These three situations are:

1. At the beginning of the gathering session, the value-driven system will not have any information sources available to it or any instances to evaluate. Thus, the only option available is to query a link source and an instance source in order to create the initial set of instances to choose from and create a set of potential information sources.
2. If the remaining set of information sources have an extremely high cost and/or low query value. In this case, the system will query the link source in order to have a broader set of potential information sources from which to choose.
3. If the system evaluates that all of the existing instances have a very low expected utility, no matter what future evidence is returned relating to them. For example, in the restaurant system, a user could have a strong preference for Chinese cuisine. If all of the existing instances return evidence that specifies that they do not serve Chinese cuisine, then the value-driven system will query

the instance source in the hopes that one of the new instances will be a Chinese restaurant.

Querying a link source by itself has no value. but we have a method for determining the value of querying an information source given the value of the information to the decision and the responsiveness of the information source. The value of querying a link source is calculated by combining the two querying steps into one atomic process: querying a link source and then using that information to query an information source is equivalent to querying an information source with a lower responsiveness. The penalty for querying this "virtual" information source accounted for by adjusting the response histogram for the information source based on the response histogram for the link and instance source needed to get the information to use the information source.

In principle, this additional time penalty should be divided among all of the information sources that query the link source makes available. In practice, I found that decreasing the time penalty by such a large amount did not accurately take the cost of time penalty into account. In the end, the system operated best when it charged the penalty to the first virtual information source returned by the link source with the full penalty and then applied no penalty to the other new available information source that became available after the link source returned.

Another improvement to the system was a method for disregarding instances that had no evidence and were equivalent to other instances. For example, after the VDIG system gathered a link source that returned ten new instances, only the first instance would be evaluated for querying until a query had been made for it. This actually resulted in a substantial speed increase for the system, so many of the instances are equivalent until some information has been returned on them.

The next subsection goes over the results of the value-driven approach compared to two other approaches. The final subsection goes over a simplified run of the restaurant information gathering system in detail.

	Coverage	Periodic	VDIG
Utility	27.3	29.1	33.5
Accuracy	47%	61%	85%
15 seconds of time for free and a utility penalty of 0.5 for each additional second			

Table 6.7 - Results of restaurant decision system

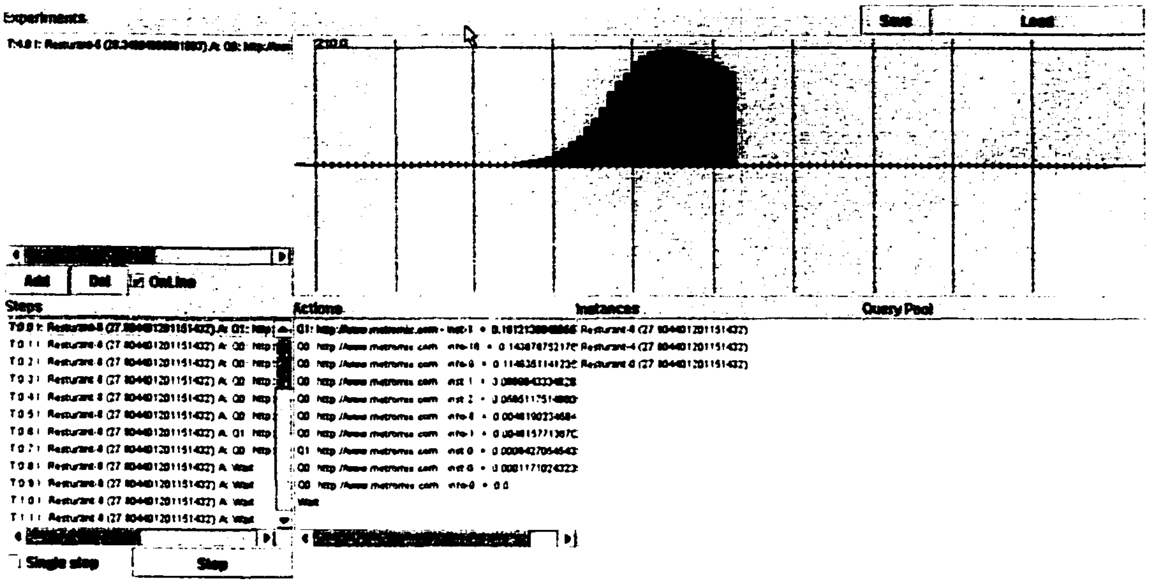
6.6.1 Domain: Restaurant selection

The value-driven system queried sites on the Internet to choose a restaurant. This decision depended on several factors: the rating of the restaurant, the cost, the location of the restaurant, the type of food the restaurant served and the preferences of the user. We focused our decision on the city of Chicago. **Figure 6.11** shows the belief network for each instance of the restaurant along with two evidence nodes from a particular query. The restaurant VDIG system also dynamically added new instances to its decision model by querying link/instance sources that returned new sets of restaurants that could be evaluated.

6.6.2 Evaluation

We ran the system continuously over several hours with different user preferences and cost functions. The results were then compared to two base-line strategies for gathering information and increasing the number of instances and information sources. The simplest approach was a coverage system where all of the information sources were given the same weight and the instance source was queried once the available number of information sources reached zero. Another approach was to query information sources by their value of information (disregarding their responsiveness) and to periodically querying the link/instances source. The performance of the periodic system is shown in **Table 6.7**.

The results of the restaurant selection system are shown in **Table 6.7**. Performance has been evaluated in a similar manner to previous experiments. We have measured the accuracy and average utility of the system compared to a number of other systems. The numerical results of the value-driven information gathering system are not drastically



```
T0.0 I: Resturant-8 (27.804401201151432) A: 01: http
T0.1 I: Resturant-4 (27.804401201151432) A: 00: http
T0.2 I: Resturant-4 (27.804401201151432) A: 00: http
T0.3 I: Resturant-4 (27.804401201151432) A: 00: http
T0.4 I: Resturant-4 (27.804401201151432) A: 00: http
T0.5 I: Resturant-4 (27.804401201151432) A: 00: http
T0.6 I: Resturant-4 (27.804401201151432) A: 01: http
T0.7 I: Resturant-4 (27.804401201151432) A: 00: http
T0.8 I: Resturant-4 (27.804401201151432) A: Wait
T0.9 I: Resturant-4 (27.804401201151432) A: Wait
T1.0 I: Resturant-4 (27.804401201151432) A: Wait
T1.1 I: Resturant-4 (27.804401201151432) A: Wait
```

```
01: http://www.metromix.com - inst-1 = 0.1612130[illegible]   Resturant-4 (27.804401201151432)
00: http://www.metromix.com  info-18 = 0.143878752170          Resturant-4 (27.804401201151432)
00: http://www.metromix.com  info-0  = 0.114826114123          Resturant-4 (27.804401201151432)
00: http://www.metromix.com  inst-1  = 0.0[illegible]
00: http://www.metromix.com  inst-2  = 0.066117514[illegible]
00: http://www.metromix.com  info-4  = 0.004819023468
00: http://www.metromix.com  info-1  = 0.004815771307
01: http://www.metromix.com  inst-0  = 0.000942706464
00: http://www.metromix.com  inst-0  = 0.000117102432
00: http://www.metromix.com  info-0  = 0.0
Wait
```

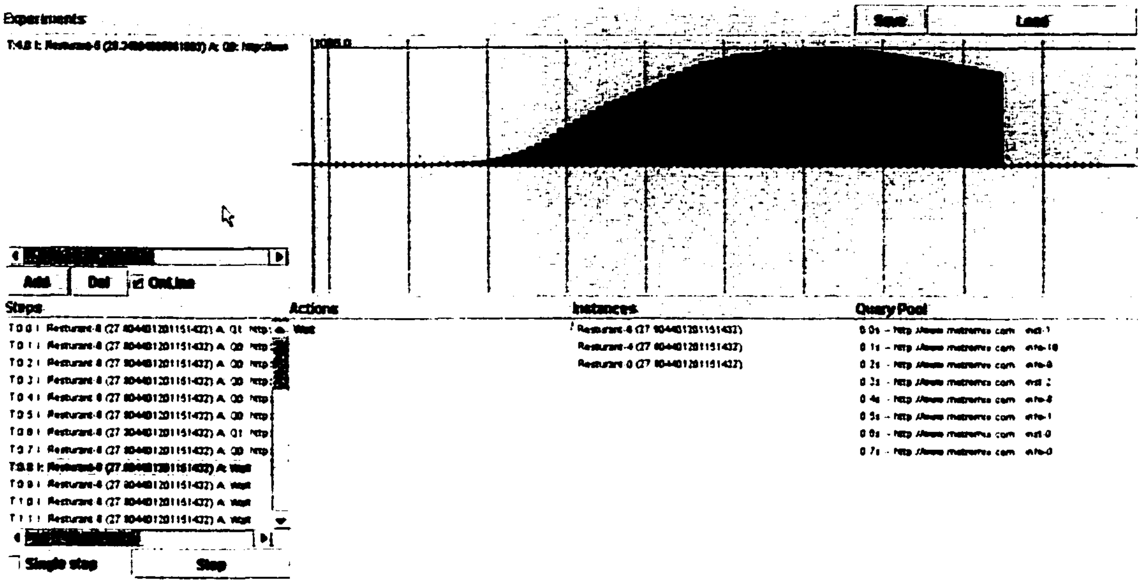
Figure 6.12 – At the start of the information gathering session

different from the results of the removable media experiment, but the amount of work needed to give the system access to hundreds of potential information sources has been greatly reduced. Another key advantage to the new value-driven system is that as MetroMix adds new sites, the value-driven system can integrate them into the gathering process without intervention from an expert.

6.7 Trace

This section shows a trace of a gathering session for the restaurant selection domain. In order for the trace to finish in a small number of steps, this is a restricted gathering session, with only five instance sources.

At the start of the value-driven process (**Figure 6.12**), the system has access to both the link/instance sources and the prototype information sources used by the link/instance sources. Remember that a prototype information source is a description of a set of information sources, which can be used to access an information source once it becomes available by querying a link/instance source. The extractors and response probability graph of the prototype information sources are cloned by the link/instance sources



T:4.8 I: Resturant-6 ([illegible]) A: Q8: http://www

T 0 0 I Resturant-8 (27 804401201151432) A Q1 http:
T 0 1 I Resturant-8 (27 804401201151432) A Q0 http:
T 0 2 I Resturant 8 (27 804401201151432) A Q0 http:
T 0 3 I Resturant 8 (27 804401201151432) A Q0 http:
T 0 4 I Resturant 8 (27 804401201151432) A Q0 http:
T 0 5 I Resturant-8 (27 804401201151432) A Q0 http:
T 0 6 I Resturant-8 (27 804401201151432) A Q1 http:
T 0 7 I Resturant-8 (27 804401201151432) A Q0 http:
T:0.8 I: Resturant-8 (27 804401201151432) A: Wait
T 0 9 I Resturant-8 (27 804401201151432) A Wait
T 1 0 I Resturant 8 (27 804401201151432) A Wait
T 1 1 I Resturant 8 (27 804401201151432) A Wait

Wait

Resturant-6 (27 804401201151432)
Resturant-4 (27 804401201151432)
Resturant 0 (27 804401201151432)

0 0s – http://www.metromix.com inst-1
0 1s – http://www.metromix.com info-10
0 2s – http://www.metromix.com info-8
0 3s – http://www.metromix.com inst 2
0 4s – http://www.metromix.com info-8
0 5s – http://www.metromix.com info-1
0 6s – http://www.metromix.com inst-0
0 7s – http://www.metromix.com info-0

Figure 6.13 – After the initial set of queries have been made

whenever new information sources are added to the information source database by a query returning from a link/instance source. The trace window is divided into six areas:

The experiment list

Contains the list of experiments that have been run. Only one is displayed here for this trace.

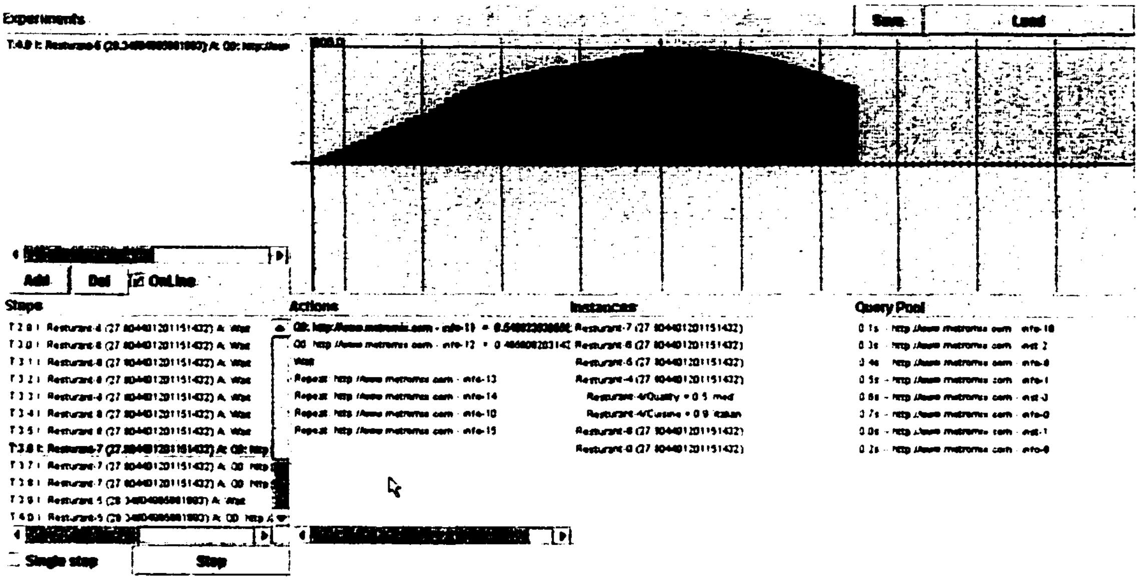
The step list

Contains the list of steps executed during this experiment. Each step has a time index, the current restaurant that would be returned at this time, and the action done during this step: either querying an information source or waiting.

The expected utility curve

Displays the expected utility curve for the experiment at the currently selected step. This is a projection into the future, so the graph does not display any information about the previous or current utility score for the session.

The action list



T 4 0 t: Resturant-5 (28.34804086881893) A: QB: http://www

Steps

T 2 0 : Resturant-4 (27 80440120115 1432) A: Wait
T 3 0 : Resturant-8 (27 80440120115 1432) A: Wait
T 3 1 : Resturant-8 (27 80440120115 1432) A: Wait
T 3 2 : Resturant-8 (27 80440120115 1432) A: Wait
T 3 3 : Resturant-4 (27 80440120115 1432) A: Wait
T 3 4 : Resturant 8 (27 80440120115 1432) A: Wait
T 3 5 : Resturant 8 (27 80440120115 1432) A: Wait
T 3 6 t: Resturant-7 (27.80440120115 1432) A: QB: http
T 3 7 : Resturant-7 (27 80440120115 1432) A: QB http
T 3 8 : Resturant-7 (27 80440120115 1432) A: QB http
T 3 9 : Resturant 5 (28 34804086881893) A: Wait
T 4 0 : Resturant-5 (28 34804086881893) A: QB http

Actions

QB: http://www.metromix.com - info-11 = 0.54802380806
QB http://www.metromix.com - info-12 = 0.48680820314
Wait
Repeat http://www.metromix.com - info-13
Repeat http://www.metromix.com - info-14
Repeat http://www.metromix.com - info-10
Repeat http://www.metromix.com - info-15

Instances

Resturant-7 (27 80440120115 1432)
Resturant-8 (27 80440120115 1432)
Resturant-6 (27 80440120115 1432)
Resturant-4 (27 80440120115 1432)
Resturant-4/Quality = 0.5 med
Resturant-4/Cusine = 0.9 italian
Resturant-8 (27 80440120115 1432)
Resturant-0 (27 80440120115 1432)

Query Pool

0.1s - http://www.metromix.com - info-10
0.3s - http://www.metromix.com - inst-2
3.4s - http://www.metromix.com - info-8
0.5s - http://www.metromix.com - info-1
0.6s - http://www.metromix.com - inst-3
3.7s - http://www.metromix.com - info-0
0.0s - http://www.metromix.com - inst-1
0.2s - http://www.metromix.com - info-6

Figure 6.14 – State of the gathering session after an information source and an instance source have returned

A list of potential actions, sorted by their *value of query* score. The top action is what the value-driven system has done during the current time step.

Instance list

A list of the instances the system is currently aware of and their expected utility value. This list is also sorted by the highest expected score to the lowest.

The query pool

A list of the queries that have been sent during the gathering session.

At step one (**Figure 6.12**), the VDIG has evaluated the set of actions which can be performed during this time step (see the actions list) and has determined that querying the instance source "inst-1" will result in the highest expected increase in utility. This is because querying the instance source will increase the set of possible instances to evaluate. At this point, the value-driven system does not have any instances that have a high expected utility, and by increasing the number of possible instances, the probability of one of these instances having a higher expected utility increases. The expected utility curve display in this figure is the expected utility curve with this action in the query pool. The decrease in the future expected utility curve is due to the cost of time function beginning to influence the overall utility.

This second screen-shot (**Figure 6.13**) shows the system after it has made its initial set of queries and is waiting for a response before making any more queries or a decision on which restaurant to select. The action list is empty except for the wait action at this step in the experiment. The instances have the same value as at step one since no evidence has been returned. The query pool at this point has eight outstanding queries. The first one fired is the query to gather a new instance (from **Figure 6.13**), and the next two are queries to prototype information sources that the system initially has in its information sources database. The system has these few initial information sources because they are used as prototypes for the information sources that will be returned by querying the link/instance sources. Each link/instance source has associated with it a set of information sources that act as prototypes for the information sources returned by the link source. For example, when the link source inst-1 returns, the VDIG system will use the same extractor that it uses for info-9 and assume that those information sources will return the same set of features with the same responsiveness as info-9. This information is part of the initial information sources database constructed by the developer of the decision making system.

At this point (**Figure 6.14**) both the inst-1 and info-9 have returned information. The link/instance source inst-9 has returned a few new restaurants to add to the instance list, and the information source info-9 has added two pieces of evidence to the belief network for restaurant-4. Unfortunately, these pieces of evidence are right in the middle of the user's preferences, and so have no effect on the preference of the user to a specific restaurant.

Also note that in the action list, a set of actions are headed with the word "repeated" these actions are not considered, because at this point they would have the same expected increase in utility as querying info-11 and info-12. Managing these redundant actions and not evaluating them has greatly increased the speed of the algorithm without any loss in performance. As explained in the previous section, many of the information

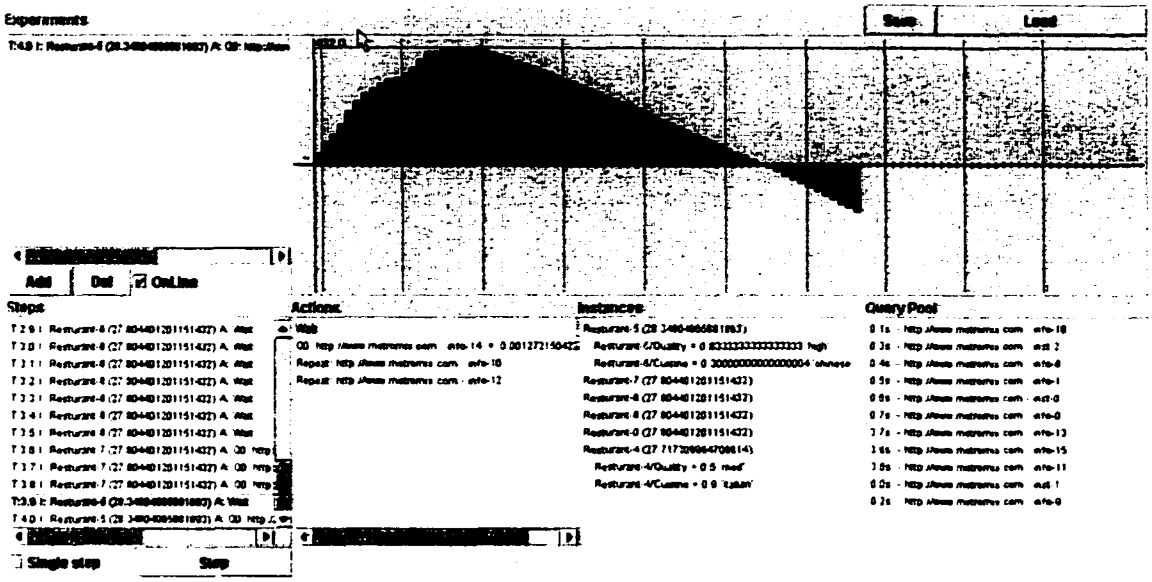


Figure 6.15 – The end of the information gathering session

sources returned by a link/instance source are identical, and only one needs to be evaluated until it is differentiated from the other identical sources (this happens once a query has been made of the information source).

At this point (**Figure 6.15**), the value-driven restaurant gathering system has neared the end of collecting information for making the decision. Although not displayed here, at the next time slice the expected utility curve collapses due to a query returning information about restaurant-5. Once this occurs, there are no more queries that will increase the overall expected utility of the gathering session.

CHAPTER 7

CONCLUSIONS

7.1 Summary of contributions

This dissertation has presented a system for creating decision making systems on the Internet that take the uncertainty and costs associated with querying external information sources into account. This has led to a number of conclusions about how to create effective decision making system that operate in complex environments (specifically the Internet):

A system for constructing internet systems based on influence diagrams can be applied to a large class of real-world problems

The experimental systems developed for this dissertation cover numerous selection and decision problems that people and organizations face constantly. Using information to make a selection from a variable set of objects can be applied in many situations. The system described in this dissertation can also be used as a starting point for other researchers, so that they may focus on specific aspects of creating more effective decision-making Internet systems.

The Internet has a large number of redundant information sources with a variety of costs and benefits

The experimental systems described in this dissertation used a number of information sources from different organizations. Section 0, 6.5.1 and 6.6.1 each use a different set of web sites as information sources. These sources were extremely varied in their cost and their benefits. One example is the information sources in the digital camera experiments: a majority of sites would contain information about several aspects of one camera, but other sources would contain information about one or two features for almost all of the cameras the system was comparing. This gave the value-driven system a particularly effective strategy of initially querying the broad sources and then using them to narrow in on a camera that best fit the user's preferences. In the restaurant selection experiments, there were so many potential choices, that no source contained information about them all. Therefore, the value-driven strategy in this case was to sample restaurants by querying sources at random until it had a small subset of high-scoring restaurants. At this point, the value-driven information gathering system would attempt to query sources that would return information about this subset.

The quality of the decision improves rapidly at the beginning of the gathering process

The removable media experiments were a good demonstration of just how quickly a value-driven approach could reach a high-quality result compared to an attempt at "coverage." A value-driven system would arrive at nearly the same quality conclusion as a system that queried the entire set of information sources in less then 25% of the time.

Influence diagrams offer a sound framework for information integration

One of the biggest questions in information extraction and integration is how to apply a sound theoretical framework to the problem. By using evidence nodes and an influence diagram as the high-level representation of the information gathering process, many of these issues are resolved. A large amount of research has been done in decision theory already, and leveraging that work into information gathering and integration is a powerful technique.

Reasoning about the resources used in information gathering on the Internet improves the effectiveness of the system

A majority of research on information gathering on the Internet does not yet consider the cost of time, extraction, or integration. Adding these considerations greatly decreases the amount of resources required while maintaining high quality decisions. As Internet systems start to become deployed more and more in the real world, these costs will become more important.

As important as the first four contributions are to constructing effective Internet systems, the final contribution of this dissertation is by far the most important. If nothing else, this dissertation has demonstrated that rational resource use by decision making systems using the Internet is vital to their effectiveness. This dissertation has also described a formal framework for addressing the problem of resource-bounded reasoning in the decision-making domain. The value-driven system described in this dissertation allows decision-making systems to develop strategies to reduce the resources used while still returning high-quality decisions.

The Internet is growing at an amazing pace; the number of available information sources for the final experimental value-driven system described in this dissertation was drastically larger than the number of information sources available for the first experimental system constructed three years earlier. Reasoning about the cost as well as the benefits of querying any potential information source will only become more important to decision-making Internet systems as the amount and variety of available information on the Internet continues to grow. Value-driven information gathering provides an effective technique for addressing these challenges.

7.2 Limitations and extensions

7.2.1 Scalability

One critical question with respect to the value-driven approach relates to its scalability. Some aspects of the system can easily scale up. These easy to scale aspects include: learning and maintaining the responsiveness of sites and increasing the number of object instances being evaluated (since each is a separate belief network that does not interact with the other networks except through the decision node). More challenging aspects include automating the discovery of high-quality information sources, the automatic construction of wrappers, learning the accuracy and bias of information sources, and increasing the complexity of the individual belief networks. Some of these challenges are being addressed by other researchers. For example, Horvitz [41][42] has developed methods for estimating the state of large-scale belief networks under resource-constraints. Koller and Pfeffer [50] have developed an object-oriented methodology for influence diagrams to facilitate their design. Moreover, several researchers have been working on methods for automatic wrapper generation [4][18].

7.2.2 Combining with other research

Other researchers in information gathering are working on complimentary technologies. Some of this work has focused on combining information from multiple sources [3][67] and developing a hierarchy of Internet resources [24]. The BIG project [56] has developed another resource-bounded approach to information gathering. What distinguishes our approach is the explicit representation of the user's decision model, which drives the information gathering process using a well-defined notion of information value.

7.2.3 Internal resource allocation

Extending model-based reason to reason about the cost of gathering information has opened a large number of possible areas of research. At this point, I have been studying the notion of independent processes that return information. In fact, the time required to translate this information into a form that is usable for instantiating the decision model is non-trivial. The internal task of information extraction from raw data differs from collecting data in that we have some control over how to prioritize the translation process. The benefits of expanding the system in this manner should be apparent, if two information sources return raw data at the same time, the system should prioritize extracting the information that will improve the decision the most. In addition, in some domains, there will be different extraction techniques that require different computational resources and return data of varying quality. This leads us back full circle to our original work in anytime algorithms, but this time as a component of a larger value-driven system.

7.2.4 Myopic planning

Value-driven information gathering has so far focused on myopically choosing the next best querying action. In the future, it might pay to consider sets of actions together to generate an entire gathering plan and then modify it if information is returned unexpectedly early or late from an information source. Until now we have considered the high level of uncertainty and high computational cost to building an entire gathering plan not worth the possible increase in utility, but there may be domains in which constructing a querying plan would offer substantial benefits to the overall gathering process.

7.3 Future directions

This work has demonstrated that there is a substantial increase in performance by constructing information gathering systems that take resource usage as well as the value of information into account. Value-driven information gathering is not a replacement for traditional information retrieval over the Internet. A value-driven system does not provide a generalized query language to describe the information requested by the user. Instead, the future of value-driven information gathering systems is in the construction of specialized high-level information gathering and decision making tasks. These high-level information gathering systems will have a more detailed description of both the domain in which they operate and the needs of the user making a request from the system. In the case of this work, that information was maintained in the decision model, the utility function, and the resource cost function.

Information retrieval systems and information gathering systems will continue to develop side-by-side as the complexity of the information accessible on the Internet and the services required by users continue to increase. Another major factor in the construction of information gathering systems will be the increase in the amount of data that is created for the use and manipulation of autonomous systems as opposed to data that is created for use by humans. One of the primary challenges facing both information retrieval and information gathering systems is that the information contained in web is not created for use by these systems. The construction of information extraction systems is a time-consuming and difficult task. As more information is created on the Internet for use by autonomous information gathering systems, the capabilities and usefulness of these systems will greatly increase.

The work that has been presented in this dissertation provides a framework for the construction of autonomous systems that use a detailed description of the decision, as well as knowledge about the use and costs of information sources to effectively gather information and make decisions.

BIBLIOGRAPHY

[1] Adelberg, Brad. Nodose - a tool for semi-automatically extracting structured and semi-structured data from text documents. In *SIGMOD Conference 1998* (1998).

[2] Ambite, Joes Luis, and Knoblock, Craig A. Flexible and scalable cost-based query planning in mediators: A transformational approach. *AI Journal Special Issue on Intelligent Internet Systems 118*, 1-2 (Apr. 2000), 115-162.

[3] Ambite, Jose' Luis, and Knoblock, Craig A. Planning by rewriting: Efficiently generating high-quality plans. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence* (Providence, RI, 1997).

[4] Ashish, Naveen, and Knoblock, Craig. Semi-automatic wrapper generation for internet information sources. In *Second IFCIS Conference on Cooperative Information Systems (CoopIS)* (Charleston, South Carolina, 1997).

[5] Ashish, Naveen, and Knoblock, Craig. Wrapper generation for semi-structured internet sources. In *ACM SIGMOD Workshop on Management of Semi-structured Data* (Tucson, Arizona, 1997).

[6] Ashish, Naveen, and Knoblock, Craig A. Information gathering plans with sensing actions. In *Proceedings of the Fourth European Conference on Planning* (Toulouse, France, 1997).

[7] Berners-Lee, T., Hendler, J., and Lassile, O. The Semantic Web. In *Scientific American*, May 2001.

[8] Birmingham, W. P., Durfee, E. H., Mullen, T., and Wellman, M. P. The distributed agent architecture of the University of Michigan digital library. In *AAAI Spring Symposium on Information Gathering in Heterogeneous. Distributed Environments* (Stanford, CA, 1995), pp. 19-24.

[9] Boddy, M., and Dean, T.L. Solving time-dependent planning problems. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence* (Detroit, Michigan, 1989), pp. 979-984.

[10] Bosak, J., and Bray, T. XML and the Second-Generation Web. In *Scientific American, May 1999.*

[11] Callan, J., Croft, W. B., and Harding, S. The inquery retrieval system. In *Proceedings of the 3rd International Conference on Database and Expert Systems Applications* (1992), pp. 78-83.

[12] Castillo, Enrique, Gutierrez, Jose Manuel, and Hadi, Ali S. *Expert Systems and Probabilistic Network Models*. Springer-Verlag, New York, 1997.

[13] Chandrasekharan, B., Josephson, John R., and Benjamins, V. Richard. What are ontologies and why do we need them? *IEEE Intelligent Systems* (Jan. 1999), 20-26.

[14] Cohen, William W. The whirl approach to integration: An overview. In *AAAI-98 workshop on AI and Information Integration* (1998).

[15] Craven, M., DiPasquo, D., Freitag, D., McCallum, A., Mitchell, T., Nigam, K., and Slattery, S. Learning to construct knowledge bases from the world wide web. *AI Journal Special Issue on Intelligent Internet Systems 118*, 1-2 (Apr. 2000), 69-114.

[16] de Finetti, B. Foresight: its logical laws, its subjective sources. In *Ann. Inst. H. Poincare* (1937), vol. 7, pp. 1-68. Translated by H. E. Kyburg.

[17] Dean, T. and Boddy, M. An analysis of time-dependent planning. In *Proceedings of the Seventh National Conference on Artificial Intelligence* (AAAI-88) (St. Paul, Minnesota, 1988). Morgan Kaufmann, pp. 49-54.

[18] Doorenbos, Robert B., Etzioni, Oren, and Weld, Danial S. A scalable comparison-shopping agent for the world-wide web. In *Proceedings of the Agents 97 Conference* (Marina del Rey, California, 1996).

[19] Doyle, J. The roles of rationality in reasoning. *Computational Intelligence 8*, 2 (May 1992), 376-409.

[20] Duschka, O., and Levy, A. Recursive plans for information gathering. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI-97)* (Nagoya, Japan, 1997).

[21] Duschka, Oliver M., and Genesereth, Michael R. Query planning in infomaster. Tech. rep., Stanford, 1997.

[22] E., Riloff, and Lehnert, W. Automated dictionary construction for information extraction from text. In *Proceedings of the ninth IEEE Conference on Artificial Intelligence for Applications* (1993), pp. 93-99.

[23] Etzioni, O., Hanks, S., Jiang, T., Karp, R.M., Madani, O., and Waarts, O. Efficient information gathering on the internet. In *Proceedings of the Symposium on Foundations of Computer Science* (Burlington, Vermont, 1996).

[24] Etzioni, Oren. Moving up the information food chain: Deploying softbots on the World Wide Web. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence* (Portland, Oregon, 1996).

[25] Etzioni, Oren, and Weld, Daniel. A softbot-based interface to the internet. *Communications of the ACM 37*, 7 (July 1994), 72-76.

[26] Fox, E. Digital libraries. *IEEE Computer 26*, 11 (Nov. 1993), 79-81.

[27] Garvey, A., and Lesser, V. Design-to-time real-time scheduling. *IEEE Transactions on Systems, Man and Cybernetics, Special Issue on Planning, Scheduling and Control 23*, 6 (Apr. 1994).

[28] Geddis, Donald F., Genesereth, Micheal R., Keller, Arthur M., and Singh, Narinder P. Infomaster: A virtual information system. In *Intelligent Information Agents Workshop* (Baltimore, Maryland, 1995).

[29] Genesereth, Michael R., Keller, Michael A., and Mueller, Glen C. Stanford information network. Tech. rep., Stanford, 1996.

[30] Gio, W. The roles of artificial intelligence in information systems. *Journal of Intelligent Information Systems 11*, 1 (1992), 35-56.

[31] Gladney, H., Belkin, N., Ahmed, Z., Fox, E., Ashany, R., and Zemankova, M. Digital library: Gross structure and requirements. In *Proceedings of Workshop on On-line Access to Digital Libraries* (1994).

[32] Good, I. J. *Probability and the weighing of evidence.* Griffin, London, 1950.

[33] Grass, J., and Zilberstein, S. From html to usable data: Problems in meaning and credibility in the www. In *AAAI-98 workshop on AI and Information Integration* (Madison, WI, 1998).

[34] Grass, J., and Zilberstein, S. A value-driven system for autonomous information gathering. *Journal of Intelligent Information Systems 14* (2000), 5—27.

[35] Hammer, J., Garcia-Molina, H., Cho, J., Aranha, R., and Crespo, A. Extracting semistructured information from the web. In *Workshop on Management of semistructured data* (1997).

[36] Hansen, E. A., and Zilberstein, S. Monitoring the progress of anytime problem solving. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence* (Portland, Oregon, 1996), pp. 1229-34.

[37] Henrion, M. Practical issues in constructing belief networks. In *Uncertainty in Artificial Intelligence* (Amsterdam, North Holland, 1984), vol. 3, pp. 161-173.

[38] Horvitz, E., and Peot, M. Flexible strategies for computing the value of information in diagnostic systems. In *Proceedings of the AAAI Fall Symposium on Flexible Computation in Intelligence Systems* (Cambridge, Massachusetts, 1996), pp. 89-95.

[39] Horvitz, E. J. Reasoning about beliefs and actions under computational resource constraints. In *Proceedings of the 1987 Workshop on Uncertainty in Artificial Intelligence* (Seattle, Washington, 1987).

[40] Horvitz, E. J., Suermondt, H. J., and Cooper, G. F. Bounded conditioning: Flexible inference for decision under scarce resources. In *Proceedings of the 1989 Workshop on Uncertainty in Artificial Intelligence* (Windsor, Ontario, 1989), pp. 182-193.

[41] Horvitz, Eric. Reasoning under varying and uncertain resource constraints. In *Seventh National Conference on Artificial Intelligence* (1988), pp. 111-116.

[42] Horvitz, Eric, and Peot, Mark. Flexible strategies for computing the value of information in diagnostic systems. In *Working notes for the AAAI-96 Fall Symposium on Flexible Computation in Intelligent Systems: Results, Issues, and Opportunities* (1996), pp. 89-95.

[43] Horvitz, Eric J., Cooper, Gregory F., and Heckerman, David E. Reflection and action under scarce resources: Theoretical principles and empirical study. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence* (Detroit, MI, 1989), pp. 1121-1127.

[44] Howard, R. A. Information value theory. *IEEE Transactions on Systems Science and Cybernetics 2*, 1 (1966), pp. 22-26.

[45] Howard, R. A., and Matheson, J. E. Influence diagrams. *Principles and applications of decision analysis 2* (1984).

[46] Huffman, S. B. Learning information extraction patterns from examples. *IJCAI-95 Workshop on New Approaches to Learning for Natural Language Processing* (Aug. 1995).

[47] Jensen, F., and Liang, J. A system for value of information in Bayesian networks. In *Proceedings of the 1994 Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems* (1994), pp. 178-183.

[48] Knoblock, Craig, Minton, Steven, Ambite, Jose Luis, Ashish, Naveen, Modi, Pragnesh, Muslea, Ion, Philpot, Andrew G., and Tejada, Sheila. Modeling web sources for information integration. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence* (Madison, WI, 1998).

[49] Knoblock, Craig A., and Levy, Alon Y. Exploiting run-time information for efficient processing of queries. In *Working Notes of the AAAI Spring Symposium on Information Gathering from Heterogeneous, Distributed Environments* (Stanford, CA, Mar. 1995), Stanford University.

[50] Koller D. and Pfeffer. A. Object-Oriented Bayesian Networks. In *Proceedings of the 13th Annual Conference on Uncertainty in AI (UAI)* (Providence, Rhode Island, August 1997), pp. 302-313.

[51] Konopnicki, D., and Shmueli, O. W3qs: A query system for the world wide web. In *Proceedings of the 21st International Conference on Very Large Databases* (Zurich, Switzerland, 1995).

[52] Kushmerick, N. Wrapper induction: Efficiency and expressiveness. *AI Journal Special Issue on Intelligent Internet Systems 118*, 1-2 (Apr. 2000), 15-68.

[53] Kwok, Chung, and Weld, Daniel. Planning to gather information. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence* (1997).

[54] Lesser, V., Horling, B., Klassner, F., Raja, A., Wagner, T., and Zhang, S. Big: A resource-bounded information gathering agent. In *Fifteenth National Conference on Artificial Intelligence* (1998), pp. 539-546.

[55] Lesser, Victor. Horling, Bryan, Klassner, Frank, Raja, Anita, and Wagner, Thomas. Information gathering as a resource bounded interpretation task. Tech. rep., University of Massachusetts, Amherst, 1997.

[56] Lesser, V.R., Horling, B., Klassner, F., Raja, A., Wagner, T., and Zhang, S.X.Q. Big: An agent for resource-bounded information gathering and decision making. *AI Journal Special Issue on Intelligent Internet Systems 118*, 1-2 (Apr. 2000). (Also available as University of Massachusetts/ Amherst CMPSCI Technical Report 1998-52.).

[57] Mahalingam, Kuhanandha. Ontology tools for semantic reconciliation in distributed heterogeneous information environments. *Intelligent Automation and Soft Computing: An International Journal (special issue on Distributed Intelligent Systems)* (1998).

[58] Pearl, J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan-Kaufmann, Los Altos, California, 1988.

[59] Perkowitz, M., and Etzioni, O. Toward adaptive web sites: Conceptual framework and case study. *AI Journal Special Issue on Intelligent Internet Systems 118*, 1-2 (Apr. 2000), 245-276.

[60] Pradhan, M., Provan, G., Middleton, B., and Henrion, M. Knowledge engineering for large belief networks. In *Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence* (Seattle, Washington, 1994), R.L. de Mantara and D. Poole, Eds., Morgan Kaufmann Publishers, pp. 448-490.

[61] Russell, S. J., and Wefald, E. H. Principles of metareasoning. In *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning* (San Mateo, California, 1989), R.J. Brachman, Ed., Morgan Kaufmann.

[62] Russell, Stuart, and Norvig, Peter. *Artificial Intelligence: A Modern Approach.* Prentice Hall, New Jersey, 1995.

[63] Russell, S.J. and Wefald, E.H. On optimal game-tree search using raticnal meta-reasoning. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence* (IJCAI-89) (Detroit, Michigan, 1989), Morgan Kaufmann, pp. 334-340.

[64] Russell, S.J. and Wefald, E.H. *Do the Right Thing: Studies in Limited Rationality.* MIT Press, Cambridge, Massachusetts, 1991.

[65] Rutkowski, A. Testimony before the U.S. house of representatives committee on science, 1996.

[66] Savage, L. J. *The foundations of statistics.* Wiley, New York, 1954.

[67] Selberg, Erik, and Etzioni, Oren. The metacrawler architecture for resource aggregation on the web. *IEEE Expert* (1997).

[68] Shachter, R. D. Evaluating influence diagrams. *Operation Research 34*, 6 (1986), 871-882.

[69] Shehory, Onn M., Sycara, Karia, and Jha, Somesha. Multi-agent coordination through coalition formation. In *Proceedings of ATAL-97* (Providence, RI, 1997).

[70] Standard, Hong Kong. Xml puts tag on communications, Feb. 2000.

[71] von Neumann, J., and Morgenstern, O. *Theory of games and economic behavior*. Princeton University Press, Princeton, 1944.

[72] Volpe National Transportation Systems Center. *Value of Information and Information Services*. Publication No. FHWA-SA-99-038, U.S. Department of Transportation, October 1998. http://www.fhwa.dot.gov/reports/viiscopv.htm.

[73] Wagner, Thomas, Garvey, Alan, and Lesser, Victor. Criteria-directed task scheduling. *International Journal of Approximate Reasoning* (1998). Special Scheduling Issue.

[74] Wagner, Thomas A., Garvey, Alan J., and Lesser, Victor R. Design-to-criteria scheduling: Managing complexity through goal-directed satisficing. In *Proceedings of the AAAI-97 Workshop on Building Resource-Bounded Reasoning Systems* (July 1997).

[75] Zilberstein, S. *Operational Rationality through Compilation of Anytime Algorithms*. PhD thesis, University of California, Berkeley, 1993. Also Technical Report No. CSD-93-743, Available on-line at http://anytime.cs.umass.edu/~shlomo.

[76] Zilberstein, S., and Russell, S. J. Anytime sensing, planning and action: A practical model for robot control. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence* (Chambery, France, 1993), pp. 1402-1407.