Efficient Query Processing for Information Gathering Agents*

Craig A. Knoblock
Information Sciences Institute and
Department of Computer Science
University of Southern California
4676 Admiralty Way
Marina del Rey, CA 90292
knoblock@isi.edu

Alon Levy
AT&T Bell Laboratories
600 Mountain Ave., Room 2C-406
Murray Hill, NJ 07974
levy@research.att.com

1 Introduction

In the not too distant future, we envision information gathering agents that will have access to a large set of diverse information sources. These information sources will not belong to the agent, rather they will be information resources that are made available across the network (possibly for a fee). The agents will not maintain any real data, they will only have descriptions of the contents of these information sources. An agent will have a domain model of its area of expertise (e.g., a class hierarchy describing the objects in it domain), and a description of an information source relates the contents of the source to the domain model of the agent. An agent will serve as a mediator for processing information requests, sending requests to the appropriate information sources and possibly processing the intermediate data, thus freeing a user from being aware of and sending queries directly to the information sources.

The characteristics distinguishing this environment from traditional knowledge base and database systems can be summarized as follows:

1. Agents do not have any real data, only descriptions of information sources that contain the data.

*The first author is supported in part by Rome Laboratory of the Air Force Systems Command and the Advanced Research Projects Agency under contract no. F30602-91-C-0081, and in part by the National Science Foundation under grant number IRI-9313993. The second author is supported by AT&T Bell Laboratories. The views and conclusions contained in this paper are the author's and should not be interpreted as representing the official opinion or policy of DARPA, RL, NSF, Bell Labs, or any person or agency connected with them.

2. An agent's knowledge about the contents of the information sources is incomplete.

3. There is a large number of information sources.

4. Information may reside redundantly in many sources.

5. Sources are autonomous.

6. Access to sources is not always possible (e.g., network failures) and may be expensive (either in time or in dollars).

7. Sources use different languages, ontologies, protocols, etc.

There are many issues that must be addressed to build agents that can intelligently operate in this environment. We have addressed some of these issues in previous work [4, 5], but one issue not previously addressed is how to efficiently process information requests when there are multiple autonomous information sources and the information about the contents of these information sources is incomplete. Traditionally, query processing systems have relied on static plans (i.e., plans created at compile time by examining only the query). While this is appropriate for environments where there is complete information about the data, we argue in this paper that it is not appropriate for information gathering agents. The following examples illustrate the problem.

- Determining the relevant information sources at compile time may not be possible. Consider the query

  $$areaCode(PaloAlto, ac) \wedge phone(BobJones, ac, number).$$

  There may be many directory services available. Just by looking at the query, it is not possible to rule any of them out, and therefore a plan generated by the query processor that retrieved both the $areaCode$ and the $phone$ information and then combined them would require an excessive amount of data. However, once we have a binding for the area code of Palo Alto, the number of information sources that will give us 415 numbers will be much smaller, and therefore we can create an effective query plan. This type of optimization has been considered for logic programs with in-memory databases, and is called sideways information passing. However, it has not been generalized to separate databases, where the tradeoffs between parallel access and sideways information passing must be considered.

- Even if some of the bindings are given (or passed sideways) it doesn't necessarily mean that the query processor can use them to prune the set of relevant sources. For example, if my query is

  $$PaperBy(RonBrachman, X)$$

  then the system still has to consider all possible paper repositories, because it doesn't know anything about Ron Brachman (e.g., field, affiliation). Note that the source of the problem is that the system does not *know* anything about the binding (except the actual value) in order to prune the relevant information sources.

In our previous work [1, 4, 5] we have shown how to statically determine the information sources that are relevant to a given query. In this paper we argue that efficient query processing in information gathering agents requires that we complement these methods with (1) interleaving planning and execution of the query, (2) exploiting additional knowledge we get about bindings in the process of finding them, and (3) actively seeking additional information about bindings in order to narrow the set of relevant information sources. Recent work by by Etzioni [3] on software agents does exploit the use of additional information gathering actions to determine what to do next. However, that work encodes these actions as part of the planning operators. In this paper we present a more general framework that identifies a variety of different types of information about bindings that can be exploited and ways in which it can be found and used.

2 A Framework for Dynamic Query Processing

We describe a general framework for dynamic query processing in information gathering agents. We identify two dimensions in the framework. The first includes the kinds of information that can be obtained at run-time and how they can be used. The second dimension identifies the ways in which this information can be obtained.

2.1 A Taxonomy of Information Types

For simplicity, we assume in our discussion that an information agent is using a frame-based representation language (e.g., LOOM [7], CLASSIC [2]), where domain objects are organized into *classes* (denoted by C, C_i etc.) and binary relations among objects are represented by *roles* (denoted by r, r_i etc.). The information agents has a set of classes and roles describing its domain of expertise.

An information source s can be viewed as providing some knowledge about a class in the domain model C_s. It can either provide *some* or *all* of the instances of the class C_s. In the latter case we will say that s is a *complete* source. The source s also provides some role fillers for the instances it knows about. Formally, s provides the role fillers for the roles $r_1^s, \ldots, r_n^s$. For each role, s may provide all the fillers or only some of them. The information about which class and roles s knows about it is contained in the description of s that is available to the agent.

We can now describe the kinds of information that can be obtained by the agent at run time and how they can be used. The first set of information types (called *domain information*) include information about the class hierarchy and individuals in those classes. Specifically, we have identified the following types of information:

Membership An individual being a member (or not a member of a class), for example, Ron Brachman being an instance of AI-researcher.

Fillers One or more individuals filling a role of another individual (or not being a filler of a role), for example, that the affiliation of Ron Brachman is AT&T Bell Labs.

Size The size of a class or the number of fillers of a role.

Constraints High level constraints on classes or fillers of roles (e.g., all fillers are in a certain range).

Relationships Relationships between different classes or roles (e.g., one class contains another). [1]

The second set of information types (called *source information*) are like the above types, but concerns knowledge about information sources, and not about the domain model's class hierarchy:

Membership An individual being found in an information sources (or not being found there).

Fillers One or more individuals filling a role of another individual in a specific information source.

Size The number of class instances found in a specific information sources.

Constraints High level constraints specific to an information source (e.g., an information source only contains Bell Labs researchers).

Relationships Relationships between different classes or roles (e.g., source s_1 containing all the data in source s_2).

It should be noted that in some cases the domain information can be inferred from the source information, and the description of the sources.

2.1.1 Usages of the Information

There are several ways in which the information types outlined above can be used to optimize queries:

Membership Membership information can be useful in identifying an information source that is likely to contain additional information. If we found the individual a in source s, and a subsequent subgoal asks for the filler of a role r of a, we will first check whether s contains fillers for r (which will be known in the description). Note that this type of information is especially useful because typically information sources will only have *part* of the instances of a class, and therefore, finding an instance in a given information sources is a significant piece of information.

Fillers Information about specific fillers for roles can be used to constrain the queries to other information sources. For example, if we learn the area code for Bob Jones from one information source, then it can be incorporated into the query sent to another information source.

---

[1] Note that intensional subsumption relationships between classes are can be inferred in the domain model. This class of information refers to extensional containment relationships, e.g., in the current state, all instances of C_1 are also instances of C_2.

Size Size information about classes and intermediate results is useful in ordering subgoals in a query. Traditional query processing systems estimate sizes before processing starts, but using actual size information may be critical when good estimates are unavailable [8].

Relationships The main use of additional domain model information is to rule out possible information sources. Knowing that an individual belongs to a more specific class that can be inferred from the query enables us to limit the number of sources considered in later subgoals of the query that contain the individual as a binding. For example, knowing that Ron Brachman is an AI researcher enables us to focus on paper repositories that provide AI publications. Knowing that he is an AT&T employee provides a justification for considering first a paper repository from AT&T researchers.

Constraints Domain-level constraints can be used by propagating the restrictions from one subgoal to the next. This is similar to some of the reformulations done with semantic query optimization, except that the constraints are identified dynamically instead of using precompiled information.

Completeness Completeness information about a class (or the fillers of a role) enable us to stop searching for more instances of the class (or fillers of that role).

2.2 Obtaining Domain and Source Information

A second dimension along which dynamic query processing methods differ is the way that the domain and source information are obtained:

- Information can be found by simply solving subgoals in the query. Instead of recording only the values of the bindings that are found in solving a subgoal, we can also record the information sources in which they are found. Additional domain knowledge can be inferred from the description of the information source in which the binding was found. For example, if Ron Brachman was found in the AAAI-fellow information source, then we can infer that he is a member of the class AAAI-fellows, which is a subclass of AI-researcher. If Brachman was not found in an information source that contains *all* physics researchers, then we can infer that he is not a physicist. This has already been implemented in the Information Manifold [6]

- Information about a binding can be found in the process of trying to solve the subgoal that needs the information. For example, we may begin considering a few paper repositories to find Brachman's papers, and by doing so figure out that he is a member of AI-researcher class. This will enable us to prune the subsequent paper repositories we consider.

- Information gained in solving previous queries can be used. The challenge here is to remember from previous queries only information that may be relevant in future queries, and will not change rapidly.

- Finally, an the information agent can create new subqueries in order to actively seek information about bindings. For example, by considering the descriptions of information sources providing paper repositories, the agent can determine that knowing the affiliation and field of an author dramatically reduce the number of relevant information sources. Therefore, the agent may first pose a query looking for Brachman's field and affiliation, before solving the query.

3 Discussion

This paper provides the first step towards building efficient information gathering agents in an environment with a large number of information resources and only limited information about their contents. We have presented a taxonomy of the different types of run-time information, described how this information can be used, and finally identified the different way in which it can be collected. We are currently working on the next step, which is to design a query processor that collects and uses the different types of information described here.

References

[1] Yigal Arens, Chin Y. Chee, Chun-Nan Hsu, and Craig A. Knoblock. Retrieving and integrating data from multiple information sources. *International Journal on Intelligent and Cooperative Information Systems*, 2(2):127–158, 1993.

[2] R. J. Brachman, A. Borgida, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Resnick. Living with CLASSIC: When and how to use a KL-ONE-like language. In John Sowa, editor, *Principles of Semantic Networks*, pages 401–456. Morgan Kaufmann, San Mateo, CA, 1991.

[3] Oren Etzioni, Steve Hanks, Daniel Weld, Denise Draper, Neal Lesh, and Mike Williamson. An approach to planning with incomplete information. In *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning*, pages 115–125, Cambridge, MA, 1992.

[4] Craig Knoblock, Yigal Arens, and Chun-Nan Hsu. Cooperating agents for information retrieval. In *Proceedings of the Second International Conference on Cooperative Information Systems*, Toronto, Canada, 1994. University of Toronto Press.

[5] Alon Y. Levy, Yehoshua Sagiv, and Divesh Srivastava. Towards efficient information gathering agents. In *Working Notes of the AAAI Spring Symposium on Software Agents, Stanford, California*, pages 64–70, 1994.

[6] Alon Y. Levy and Divesh Srivastava. Data model and query evaluation in the information manifold system. In preparation, 1994.

[7] Robert MacGregor. A deductive pattern matcher. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, Saint Paul, Minnesota, 1988.

[8] David E. Smith and Michael R. Genesereth. Ordering conjunctive queries. *Artificial Intelligence*, 26(2):171–215, 1985.