Information Gathering Plans With Sensing Actions *

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

Alon Levy
AT&T Bell Laboratories
AI Principles Research Dept.
600 Mountain Ave., Room 2A-440
Murray Hill, NJ 07974
levy@research.att.com

Abstract

Information gathering agents can automate the task of retrieving and integrating data from a large number of diverse information sources. The key issue in their performance is efficient query planning that minimizes the number of information sources used to answer a query. Previous work on query planning has considered generating information gathering plans solely based on compile-time analysis of the query and the models of the information sources. We argue that at compile-time it may not be possible to generate an efficient plan for retrieving the requested information because of the large number of possibly relevant sources. We describe an approach that naturally extends query planning to use run-time information to optimize queries that involve many sources. First, we describe an algorithm for generating a *discrimination matrix,* which is a data structure that identifies the information that can be sensed at run-time to optimize a query plan. Next, we describe how the discrimination matrix is used to decide which of the possible run-time sensing actions to perform. Finally, we demonstrate that this approach yields significant savings (over 90% for some queries) in a real-world task.

[*] The first and second authors is supported in part by Rome Laboratory of the Air Force Systems Command and the Advanced Research Projects Agency under contract no. F30602-94-C-0210, and in part by the National Science Foundation under grant number IRI-9313993. 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 ARPA, RL, NSF, AT&T Labs, or any person or agency connected with them.

1 Introduction

Information gathering agents are programs that answer queries using a large number of diverse information sources (e.g., sources on the Internet, company-wide databases). These information sources do not belong to the agent, instead they are provided by autonomous sources, possibly for a fee. An information agent does not maintain any real data, but only has *models* of the contents of the available information sources. An agent has a domain model of its area of expertise and a description of how the contents of an information source relate to the classes in the domain model of the agent. User queries are posed using the domain model of the agent. Given a query, an agent serves as a mediator between the user and the information sources, by decomposing the query, sending requests to the appropriate information sources, and possibly processing the intermediate data. As such, the information agent frees the user from being aware of and sending queries directly to the information sources.

Several characteristics of this problem require us to develop solutions beyond those considered traditionally in knowledge and database systems. The main characteristic is that the number of information sources will be very large, and the agent has only the models of the sources in determining which ones are relevant to a given query. In addition, some information will reside redundantly in many sources, and access to sources will not always be possible (e.g., network failures) and may be expensive (either in time or in money).

To answer queries efficiently, an information agent must be able to determine precisely which information sources are relevant to a given query, and to retrieve as little data as possible from the relevant sources. Previous work on information agents (e.g.,[Knoblock and Levy, 1995], SIMS [Arens *et al.*, 1996, Knoblock *et al.*, 1994], the Unix Softbot [Etzioni and Weld, 1994, Etzioni *et al.*, 1994], the Information Manifold [Levy *et al.*, 1996], Occam [Kwok and Weld, 1996]) has largely focused on determining relevant sources by analyzing the query and the models of the information sources. However, as the following example shows, it is not always possible to significantly prune the set of information sources without gathering some additional information that was not explicitly mentioned in the query. Suppose we are given a query asking for the phone number of John McCarthy:

$$\mathsf{Person(x)} \wedge \mathsf{Name(x, ``JohnMcCarthy'')} \wedge \mathsf{PhoneNumber(x, number)}.$$

We have access to a large number of name server repositories that contain information about individuals, email addresses, phone numbers, etc. These repositories are usually organized by institution, but without any additional information about John McCarthy this query will require accessing every repository until the information is found. In some cases the query processor can add *new* subqueries into the query plan in order to obtain additional information to constrain the sources that need to be considered. For example, the query processor could issue the subquery $\mathsf{Affiliation(``John\ McCarthy", org)}$ to the various professional organization databases to find the organization he is currently affiliated with. Using this information, the query processor can now go directly to the relevant name servers.

This example illustrates two points. First, we can obtain at run-time additional information about *individuals* or *classes* of individuals appearing in a query. The second point is that the information is obtained by *adding* discriminating queries to the original query. There are other kinds of information that can be obtained at run-time. We can obtain information about a *source* that enables us to refine our model of that source. Or, the fact that we found an individual in a *specific* source can be used later to find additional facts about this individual.

In this paper we focus on the problem of finding additional information about individuals or classes of individuals by adding discriminating queries, called *sensing actions,* to the original query. We describe a novel approach to query planning that can exploit sensing actions to optimize information gathering plans. A key component of the approach is a data structure, the *discrimination matrix*, that enables the query planner to evaluate the utility of additional sensing actions. Informally, the discrimination matrix tells us how many information sources we may be able to prune if we perform a proposed sensing action, which allows us to estimate the cost of plans with additional sensing actions. We show how to build the discrimination matrix efficiently and how to use it in query planning. Finally, we present experimental results that demonstrate that this approach produces savings of over 90% in a real-world task.

2 The Domain Model and Source Models

An information agent has a representation of its domain of expertise, called the *domain model*, and a set of *source models*, i.e., descriptions of information sources. In our discussion, we use a KL-ONE type knowledge representation language. Such a language contains unary relations (called *concepts*) which represent classes of objects in the domain and binary relations (*roles*) which describe relationships between objects. In our discussion, *complex concepts* are built from primitive concepts using the following set of constructors: (A denotes a concept name, C and D represent complex concepts, and R denotes a role):

$C, D \rightarrow A \mid$
$C \sqcap D \mid C \sqcup D \mid$ (conjunction, disjunction)
$(R < a) \mid (R > a)$ (range constraints)
$(\geq\ n\, R) \mid (\leq\ n\, R)$ (role-filler cardinality restrictions)
$(\mathsf{fills}\ R\ a)$ (filler restriction)
$(\mathsf{oneOf}\ R\ \{a_1, \ldots, a_n\})$ (restriction on fillers)

As an example, the concept Person ⊓ (fills occupation research) ⊓ (oneOf affiliation {ISI, Stanford}) ⊓ (≥ 3 student) represents the set of people, whose occupation is research, affiliated with either ISI or Stanford, and having at least 3 students.

The agent also has models of the external sources. A description of a source has the form $(D_S, r_1^s, \ldots, r_n^s)$, meaning that S has instances of the concept D_S, and for these instances it contains role fillers of the roles $r_1^s, \ldots, r_n^s$. Note that S does not necessarily contain *all* instances of D_S or all the role fillers.

Figure 1 shows some information sources that could be used to answer queries about technical reports. The most recent source for each organization would be modeled as follows:

1. ((TechReport ⊓ (fills affiliation "Carnegie Mellon) ⊓ (≥ year 1990)),authors, title, year, department, abstract)

2. ((TechReport ⊓ (fills affiliation "Stanford) ⊓ (fills department "Computer Science")), authors, title, year, report)

3. ((TechReport ⊓ (fills affiliation "AT&T Labs") ⊓ (= year 1996)), authors, title, year)

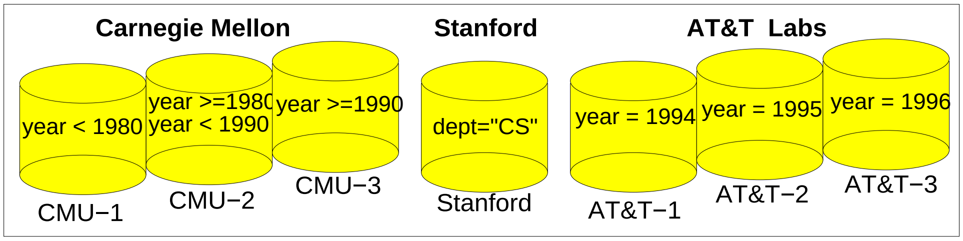

Figure 1: Example Technical Report Repositories

The first source provides the authors, title, year, department and abstract for Carnegie Mellon technical reports since 1990. The second source provides authors, title, year, and text of the report for Stanford Computer Science technical reports.

Given a query, the information agent needs to determine which information sources are relevant to it. In previous work [Etzioni and Weld, 1994, Arens *et al.*, 1996, Levy *et al.*, 1996], it was shown how to determine the relevant information sources based on the query and the models of the information sources. As a simple example, suppose our query is TechReport(x) $\wedge$ Author(x,"John McCarthy") $\wedge$ Year(x,1996). The system can infer from the query that only 3 of the 7 technical report repositories need to be considered since the other ones only contain technical reports from other years. Similarly, if the query was further specialized to include affiliation or department the set of possible sources could be pruned further. The remainder of this paper will describe techniques for performing this type of pruning at run time when relevant constraints are not explicitly specified in the query.

3 Building a Discrimination Matrix

Our approach to actively seeking information about objects involves considering query plans that contain one or more additional subqueries, called *discriminating queries*. Such queries have the form $R(\alpha, Y)$, where α is a constant and Y is the result of the discriminating query. The values obtained for Y will be used to prune the information sources considered for the subsequent subqueries in the query, because some sources will become irrelevant after obtaining new information about α.[1] Adding discriminating queries changes the cost of the query plan in two ways. First, it adds the cost of solving the additional subquery, and second, it enables us to prune the sets of the information sources relevant to subsequent subqueries. The key to evaluating the utility of the added queries, is to determine how the relevant set of information sources can be pruned when we obtain new information. The main challenge is to estimate which information sources will be pruned *without* actually knowing the values returned for $R(\alpha, Y)$.

Recall that each source S contains instances of a given class D_S, and fillers for some roles of those instances. Suppose we are searching for the fillers of role P of an individual a, and the query entails that a is a member of concept C. In this case, we will consider every information source S, such that $D_S \sqcap C$ is non-empty (i.e., is a satisfiable concept description), and such that S provides fillers for P. By obtaining the value b of the R-filler of a from a sensing action,

[1] Discriminating queries are not treated exactly as ordinary ones. Specifically, if a discriminating query fails (i.e., no values are found for Y), then that only means that we could not obtain additional information, and therefore we need to consider the subsequent subqueries without pruning any of the sources relevant to them.

we can further restrict the relevant sources to those for which $D_S \sqcap (\mathsf{fills}\, R\, b)$ is non-empty. The discrimination matrix of a role R and class C tells us which information sources would be relevant if we *also* knew the value of the filler of R of the object a.

Formally, the discrimination matrix for a role R and class C provides two kinds of information. First, the matrix partitions the possible values of R into *regions*, such that two values of R in the same region will have the same set of relevant information sources. Second, for every region r, the discrimination matrix tells us which information sources contain instances of C, given that the value of R falls within r. Intuitively, a role R provides good *discrimination* if R partitions the set of relevant information sources evenly over many regions.

Consider the example of repositories for technical reports. Building the discrimination matrix for the role year and class TechReport yields the following regions and partition of information sources:

Region	**relevant sources**
< 1980	CMU-1, Stanford
$[1980, 1990)$	CMU-2, Stanford
$[1990, 1994)$	CMU-3, Stanford
$[1994, 1994]$	CMU-3, Stanford, AT&T Labs-1
$[1995, 1995]$	CMU-3, Stanford, AT&T Labs-2
$[1996, 1996]$	CMU-3, Stanford, AT&T Labs-3
> 1996	CMU-3, Stanford

Therefore, if we know the year of the technical report for which we are searching, the number of relevant sources will be at most 3, instead of 7.

Below we describe an algorithm for constructing the discrimination matrix. The discrimination matrix is persistent and is modified only when an information source is added or deleted. In particular, it is built at compile-time, based on the source descriptions, before answering any queries. We distinguish roles whose range values are numeric from those whose range is non-numeric. In what follows we describe an algorithm for creating a discrimination matrix for numeric roles, that use constraints specified by the constructors $<$ and $>$ to partition the information sources. We then describe an algorithm for non-numeric roles, using the constructors fills and oneOf. We focus on the above constructors because these are the most likely to yield discrimination between sources.

3.1 Numeric Roles

We denote by $\mathcal{A}$ the set of constants that appear in constraints involving $<, \leq, >, \geq$ in the models of the information sources. The regions found by the discrimination matrix are the possible (open and closed) intervals involving two consecutive constants in $\mathcal{A} \cup \{\infty, -\infty\}$.

The algorithm, shown in Figure 2, builds a vector $\mathcal{V}$ of triplets (a_i, S_i^+, S_i^-). The first element in every tuple in the vector a_i is a pair of the form (n_i, θ), where n_i is a number in the set $\mathcal{A} \cup \{-\infty, \infty\}$, and θ is either $=$ or $>$. S_i^+ are S_i^- are sets of information sources. The set S_i^+ are the information sources that *become* relevant when the value of R is greater than n_i if θ_i is $>$, and greater or equal to n_i if θ_i is $=$. The set S_i^- are the information sources that *become* irrelevant when the value of R is n_i if θ_i is $=$, or become irrelevant when the value of R is greater than n_i, when θ_i is $>$.

```
procedure compute-numeric-matrix($\mathcal{S}$)
// $\mathcal{S}$ is a set of information sources
Begin with the vector $\mathcal{V} = \{((-\infty, >), \emptyset, \emptyset), ((\infty, =), \emptyset, \emptyset)\}$.
for every source $S \in \mathcal{S}$, do:
    Let $D_S$ be the class description in the model of $S$.
    Let $a_{low}$ be the largest number for which
        $D_S \models (R \ \theta \ a_{low})$, where $\theta \in \{>, \geq\}$.
    Let $a_{high}$ be the smallest number for which
        $D_S \models (R \ \theta \ a_{high})$, where $\theta \in \{<, \leq\}$.
        if $D_S \models (R > a_{low})$ then
            add-to-vector($\mathcal{V}$, $((a_{low}, >), S, \emptyset)$).
            else add-to-vector($\mathcal{V}$, $((a_{low}, =), S, \emptyset)$).
        if $D_S \models (R < a_{high})$ then
            add-to-vector($\mathcal{V}$, $((a_{high}, =), \emptyset, S)$).
            else add-to-vector($\mathcal{V}$, $((a_{high}, >), \emptyset, S)$).
return $\mathcal{V}$.
end compute-numeric-matrix.

procedure add-to-vector($\mathcal{V}$, $(a, S^+, S^-)$)
// $\mathcal{V}$ is a vector of the form $((a_1, S_1^+, S_1^-), \ldots, (a_n, S_n^+, S_n^-))$.
if $a$ is the first argument in the $i$'th element of $\mathcal{V}$ then
    $S_i^+ = S_i^+ \cup S^-$.
    $S_i^- = S_i^- \cup S^-$.
else add $(a, S^+, S^-)$ to $\mathcal{V}$ so that the first elements of each
        triplet are ordered as follows:
    $a_i = (n_i, \theta_i)$ comes before $a_j = (n_j, \theta_j)$ if
        $n_i < n_j$ or
        $n_i = n_j$, and $\theta_i$ is $=$ and $\theta_j$ is $>$.
end procedure add-to-vector.
```

Figure 2: Building a matrix for numeric roles.

The discrimination matrix is constructed from $\mathcal{V}$ as follows. Consider for example the region $[a_i, a_k)$, where $a_i, a_k \in \mathcal{A}$. The sources in the partition of this region are those that appear in the set S_j^+ for an a_j that would come before $(a_i, >)$ in $\mathcal{V}$, and do not appear in the set S_j^- for an a_j that would come before $(a_i, >)$ in $\mathcal{V}$.

Example 1: Consider the technical report repositories. We begin with the vector

$\{((-\infty, >), \emptyset, \emptyset), ((\infty, =), \emptyset, \emptyset)\}$.

Inserting the first Carnegie Mellon repository will result in the vector:

$\{((-\infty, >), CMU_1, \emptyset), ((1980, =), \emptyset, CMU_1), ((\infty, =), \emptyset, \emptyset)\}$

Inserting the other Carnegie Mellon repositories will yield the vector:

$\{((-\infty, >)CMU_1, \emptyset),$ $((1980, =), CMU_2, CMU_1),$ $((1990, =), CMU_3, CMU_2),$
$((\infty, =), \emptyset, CMU_3)\}$.

Adding the Stanford repository only changes the first and last elements of the vector:

$\{((-\infty, >), \{CMU_1, Stanford\}, \emptyset),\ ((1980, =), CMU_2, CMU_1),\ ((1990, =), CMU_3, CMU_2),$
$((\infty, =), \emptyset, \{CMU_3, Stanford\})\}.$

Finally, adding the AT&T Labs repositories will yield the following vector:

$\{((-\infty, >), \{CMU_1, Stanford\}, \emptyset), ((1980, =), CMU_2, CMU_1),\ ((1990, =), CMU_3, CMU_2),$
$((1994, =), ATT_1, \emptyset),\ ((1994, >), \emptyset, ATT_1),\ ((1995, =), ATT_2, \emptyset),$
$((1995, >), \emptyset, ATT_2),\ ((1996, =), ATT_3, \emptyset),\ ((1996, >), \emptyset, ATT_3),$
$((\infty, =), \emptyset, \{CMU_3, Stanford\})\}.$

Inserting an information source into the matrix can be done in time $O(log(n))$ in the number of information sources, n. Therefore, the overall running time of the algorithm is $O(nlog(n))$. Since the determination of the regions of R requires that we sort the values in $\mathcal{A}$, it is clear that $O(nlog(n))$ is also a lower bound on the time to build the discrimination matrix.

3.2 Non-numeric Roles

In the algorithm below assume that the domain model entails that the role R can only have a *single* filler. In the case that R may have multiple fillers, the algorithm is the same, except that we ignore the fills constructor.[2] Suppose we have a constant a in our query, and we found that b is the filler of the role R of a. This can affect the relevance of an information source S in two ways. If the description of S says that it contains instances of the class D_S, then

1. If D_S entails (fills R a), then S will be relevant if and only if a = b.
2. If D_S entails (oneOf R $\{a_1, \ldots, a_m\}$), then S will be relevant if and only if b $\in \{a_1, \ldots, a_m\}$.

Hence, if $\mathcal{A}$ is the set of constants that appear in fills and oneOf constraints in the models of the information sources, then the possible values of R can be classified into $|\mathcal{A}| + 1$ regions: one region for every value in $\mathcal{A}$ and one region for all values not mentioned in $\mathcal{A}$. Informally, the algorithm, shown in Figure 3, creates a hierarchy of *labels*. Each label denotes a subset of the regions of R. With each label L, the algorithm associates a set of information sources $Sources(L)$. An information source S will be in $Sources(L)$ if the description of S entails that the value of R *must* be in one of the regions in L. Given a set of information sources $\mathcal{S}$, we use the discrimination matrix to partition them as follows. The set of information sources in the partition of a value r are those sources in $\mathcal{S}$ that appear in the set associated with *some* label that includes r.

Consider building discrimination matrix for the role affiliation in our example. The algorithm will consider each of the available name servers in turn, and create the labels $\{MIT\}$, $\{Cornell\}$, $\{IBM\}$, $\{AT\&TLabs\}$, etc, each with one associated source.

It should be noted that although the number of possible labels that can be generated is exponential in the number of constants in $\mathcal{A}$, the algorithm will only generate at most a number

[2] The fills constructor is useful for discrimination only if we have an upper bound on the number of fillers of R.

```
**procedure compute-non-numeric-matrix**($\mathcal{S}$)
/* $\mathcal{S}$ is a set of information sources */
Initialize the discrimination matrix with the label $L^1$ =TOP and $Sources(L^1) = \emptyset$.
**for** every source $S \in \mathcal{S}$ whose associated class is given by the description $D_S$ **do:**
 **if** $D_S \models$ (ONEOF R $\{a_1, \ldots, a_n\}$) **then**
   **if** there exists a label $L = \{a_1, \ldots, a_n\}$ **then** add $S$ to $Sources(L)$.
     **else** create a new label $L = \{a_1, \ldots, a_n\}$ with
       $Sources(L) = \{S\}$ and place $L$ in the hierarchy.
 **if** $D_S \models$ (FILLS R B) **then**
   **if** there exists a label $L = \{b\}$ **then** add $S$ to $Sources(L)$.
     **else** create a new label $L = \{b\}$ with $Sources(L) = \{S\}$,
     and place $L$ in the hierarchy.
 **if** $D_S$ does not entail any such constraints on the fillers of $R$
   **then** add $S$ to $Sources(L^1)$.
**end compute-non-numeric-matrix.**
```

Figure 3: Creating a discrimination matrix for non-numeric roles. TOP is the set of all regions of R.

of labels as the number of information sources.[3] Therefore, since adding a new information source to the matrix can be done in time $O(log(n))$, where n is the number of information sources, the overall complexity of the algorithm is $O(nlog(n))$. The algorithm for the numeric roles has the same computational complexity. Note that complexity of the entailment checks done in the algorithm may be exponential in the size of the concept hierarchy [Borgida and Patel-Schneider, 1994], but not in the number of information sources.

4 Using the Discrimination Matrix

In the previous section we described how to perform the required preprocessing of a domain to identify the potentially useful discriminating queries. In this section we first describe the space of possible plans that include discriminating queries, and we then show how to use the discrimination matrix to evaluate the cost of plans.

4.1 Planning with Discriminating Queries

In the general framework described in this paper, a user issues a query in terms of the domain model and the system then determines how to efficiently retrieve the requested data. This requires selecting an appropriate set of sources to query, finding an efficient ordering of the queries, and determining what operations to perform on the data to produce the required result.

[3] Furthermore, note that if the number of labels created is close to the number of sources, that would actually mean that R is providing good discrimination.

Space does not permit a detailed review of these algorithms, but earlier work describes these algorithms in detail [Levy *et al.*, 1996, Arens *et al.*, 1996, Knoblock, 1995, Knoblock, 1996].

We consider a new space of plans in which additional discriminating queries are considered whenever there is a corresponding entry in the precomputed discrimination matrix. This new space can include plans with single discriminating queries, multiple discriminating queries, and even recursive discriminating queries. The number of additional plans considered in the new space will be relatively modest, since there are likely to be very few roles that can provide good discrimination for a given subquery in the query. The roles that do not provide good discrimination can be eliminated from consideration early on (e.g., eye color will not provide good discrimination on phone numbers, even though it is a role of class `person` since people are rarely organized by eye color).

We can extend any algorithm for searching the original set of plans as follows. Given any plan P from the original space, we consider all plans in which one or more discriminating queries are added to P immediately prior to the subqueries for which they are used to discriminate. This process is repeated until there are no additional discriminating queries to consider. The query processor would then select from among the possible plans the one with the lowest estimated cost.

Consider the problem of planning queries to the CIA World Factbook[4]. The Factbook is organized as a set of 267 Web pages, one for each country in the world, and contains information about each country's geography, population, government, etc. Queries about a specific country can be executed quickly, but queries that involve accessing large numbers of countries can take a long time because of the large number of pages that must be retrieved. Related information that can be used for discriminating queries is available in a variety of sources, such as the Yahoo Region Information[5], which provides data about the organization of countries into regions and the NATO Homepage[6], which provides a list of the current NATO countries.

Since all of the sources provide data about countries, the domain model contains the top-level class `Country` and subclasses that correspond to the individual sources. Each of the 267 countries is modeled as a separate class, each of which is a subclass of `Country` distinguished by the name of the country. There are also subclasses that correspond to different regions (i.e., Europe, Pacific Rim, etc.) and organizations (i.e., NATO, WHO, etc.). The user can ask queries about a single country, a subclass of countries, or all of the countries.

The use of discriminating queries in the Factbook can greatly reduce the number of sources that would need to be accessed. Without the use of discriminating queries, a plan for answering a query for the average GNP of all NATO countries will require accessing all 267 pages from the Factbook. Figure 4 shows the plan produced when the system cannot exploit discriminating queries. With the use of discriminating queries, the discrimination matrix would show that first determining the names of the required countries could greatly reduce the number of pages from the Factbook that would need to be retrieved. Figure 5 shows this resulting plan, where the system first performs a sensing action to retrieve the names of all of the NATO countries. Then it uses this information to generate the subplan that only requires retrieving the 16 pages from the Factbook that correspond to the NATO countries.

[4] (http://www.odci.gov/cia/publications/nsolo/wfb-all.htm)

[5] (www.yahoo.com /Regional/Regions/)

[6] (www.nato.int/family/countries.htm)

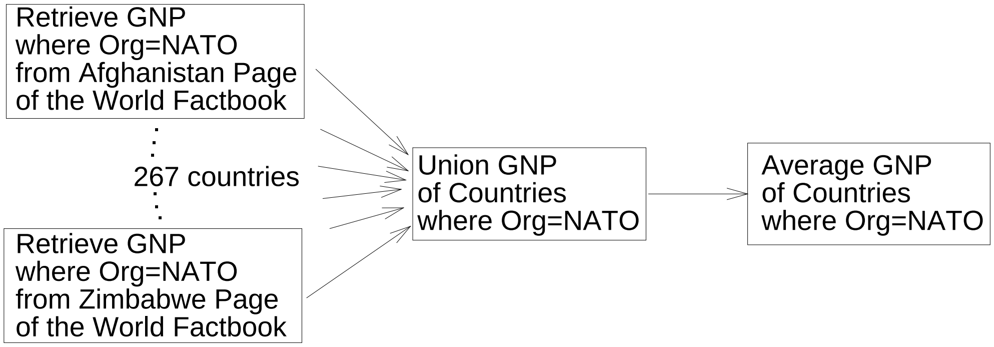

Figure 4: Original Plan Without Discriminating Query

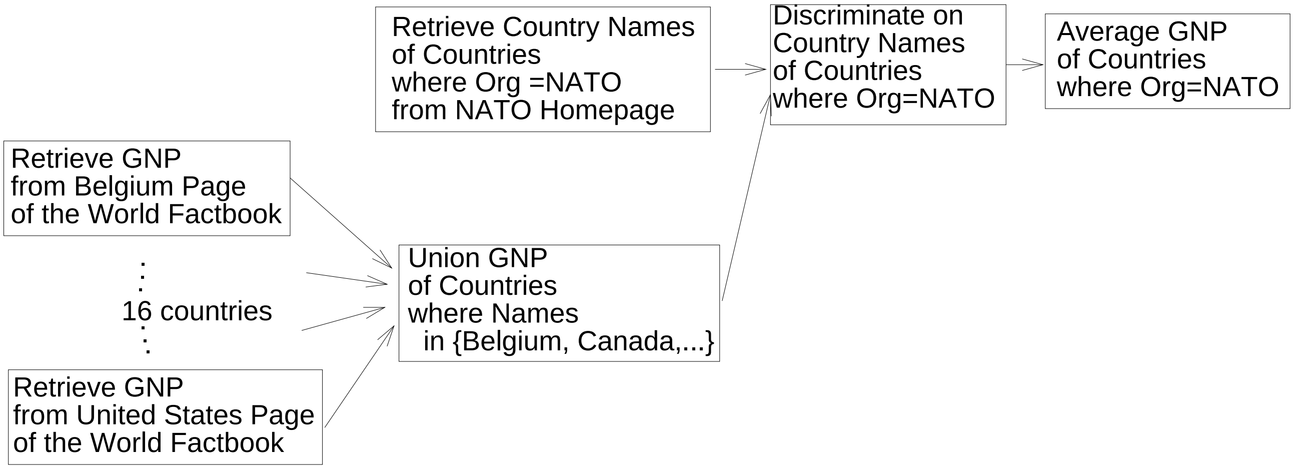

Figure 5: Plan with a Discriminating Query

4.2 Evaluating Discriminating Queries

In this section we describe how the discrimination matrix is used to evaluate the cost of a plan that includes discriminating queries. Suppose g is a subquery in the query, and we added the discriminating query $R(\alpha, Y)$ before g, where α is a constant that appears in g and Y is the result of the discriminating query. The key to evaluating the utility of the discriminating query is to determine how it will affect the set of sources relevant to g. Suppose that the query entails that α is a member of class C (e.g., TechReport, Person, or Country in our examples). The discrimination matrix for R and C tells us which sources are relevant in each possible region of R. Suppose $\mathcal{I}_1^i, \ldots, \mathcal{I}_m^i$ is the partition given by the discrimination matrix. Since we do not know in *which* of the regions the answer to the discrimination query will be, we assume the worst-case, i.e, the case that results in the maximal cost. If the combined cost of solving the discriminating query and the savings obtained for g are lower than the original cost of solving Q, then we have saved some work. The advantage of a worst-case estimate is that it guarantees that we will never spend more time gathering additional information than the time to simply execute the query against all relevant sources. We could also consider the average case if we were willing to accept that in some cases the use of a discriminating query could end up being more expensive.

In the example that requested the technical reports for “John McCarthy” there are several possible discriminations on technical reports. The query processor would consider discriminating the technical report repositories on year, organization, and both year and organization. Based on the cost and availability of this discriminating information and the amount of discrimination provided, the query processor would select the most promising plan.

The discrimination matrix can also be used to partition the sources given *multiple* discriminating queries about an object a. To determine the discrimination achieved by multiple discriminating queries, we simply take the cross products of the regions given by each query. For example, suppose the analysis of the query shows that there are six potentially relevant information sources, $\mathcal{S} = S_1, \ldots, S_6$, for retrieving the information for a query Q_i. We have two discrimination matrices, one for role R_1 and one for R_2, each having three regions. Suppose that for the role R_1 we obtain the partition $\{\{S_1, S_2\}$, $\{S_3, S_4, S_5\}$,$\{S_6\}\}$, and for R_2 the partition is $\{\{S_1\}$,$\{S_2, S_3\}$,$\{S_4, S_5, S_6\}\}$. Taking the cross products of these partitions to compute the discrimination that would be obtained by finding both R_1 and R_2, would yield nine regions, the non-empty ones being:[7]

$$\{S_1\}^{1,1}, \{S_2\}^{1,2}, \{S_3\}^{2,2}, \{S_4, S_5\}^{2,3}, \{S_6\}^{3,3}.$$

Note that the partitions for R_1 and R_2 mean that if the value for one of them is known, then at most three sources would be relevant. If both R_1 and R_2 are known, then at most two sources are relevant.

5 Experimental Results

This section presents empirical results to demonstrate that the use of sensing actions can significantly reduce the number of sources that must be accessed. To evaluate the use of sensing

[7] The superscripts denote the original regions from R_1 and R_2 respectively.

actions, we implemented the algorithms for building the discrimination matrix and extended the query planner developed by Knoblock [1995] to generate sensing actions. We then applied the resulting system to the problem of retrieving data from the CIA World Factbook.

We wrote a small set of queries to demonstrate the usefulness of the discriminating queries in this domain. The system generates the discrimination matrix for this domain model in 0.067 seconds of CPU time. Table 1 shows the time required for answering a small set of test queries both with and without sensing actions. In the first query, without sensing actions, the time required to answer the query is very high (more than an hour) as the system has to access information about each of the 267 countries to determine which countries are members of NATO. However with sensing actions, the planner decides to first fetch the names of all NATO countries (this is the sensing action) and then proceeds to fetch the defense expenditure of all the NATO countries. The agent needs to access information only about the 16 NATO member countries, which provides a significant reduction in execution time as the same query can now be answered in less than five minutes. A similar optimization is done in queries 2 and 3 where the sensing actions are to find the lists of European countries, and countries in the Pacific Rim respectively. In query 4, however, no sensing action is generated as the number of information sources to be accessed is already small (one source) in the plan without sensing actions. Thus an additional sensing action is not really expected to provide information that can further prune the number of sources to be accessed.

Query	*Time w/o sensing*	*Time w/ sensing*	*Percent Reduction*
1. Find the total defense expenditure of NATO countries	3793 sec.	287 sec.	92.4%
2. Find the average GNP of all European countries	3823 sec.	518 sec.	86.5%
3. Find all the languages spoken in the Pacific Rim.	3710 sec.	289 sec.	92.2%
4. Find the population of the United States.	14 sec.	NA	NA

Table 1: Experimental Results on CIA World Factbook Queries

6 Related Work

Our work can be viewed as a form of semantic query optimization (SQO) [King, 1981, Chakravarthy *et al.*, 1990], where new subqueries are added to a query by analyzing the integrity constraints known about the data. In our context, the analogue of integrity constraints are the descriptions of the information sources. One key issue in SQO is determining when the additional subqueries that are introduced actually lead to better performance. A contribution of our work is to provide a method for evaluating the saving achieved by the new subqueries, by estimating the number of information sources that would be accessed. Our method relies on using the models of the information sources, which have no analogue in traditional databases.

The work on Softbots for Unix by Etzioni et al. [Etzioni and Weld, 1994] exploits the use of additional sensing actions to determine where to locate information. The sensing actions are in the context of a software agent that can manipulate the environment as well as gather information. However, the specific sensing goals are encoded as explicit preconditions of the planning operators. In this paper we present a more general information gathering framework for automatically inserting useful sensing goals.

This work is also related to a variety of work on sensing in planning. In terms of the taxonomy of different uses of sensing proposed by Olawsky and Gini [Olawsky and Gini, 1990], our approach to sensing corresponds to deferring planning decisions that depend on sensor readings (i.e., the choice of which sources to query). Much of the work on sensing has focused on introducing sensing actions when there is insufficient knowledge about a problem to proceed [Olawsky and Gini, 1990, Ambros-Ingerson, 1987, Etzioni *et al.*, 1994]. In contrast, the discriminating queries exploit sensing for optimization. Finally, the previous work by both Knoblock [1995] and Levy [Levy *et al.*, 1996] presents more limited types of sensing for information gathering that exploit subqueries already in a query, but do not introduce new subqueries.

7 Limitations

There are two potential limitations to the use of discriminating queries in information gathering plans. First, the paper presented an approach to *automatically* obtaining the additional data from other sources, but there may not be another pre-existing source that can provide the required data. An alternative approach is to ask the user to provide the information about a query. This would provide an important user interface capability, where the system can effectively determine precisely what information is needed from the user in order to process a query efficiently. Another alternative is to use the techniques for building a discrimination matrix, to precompile a set of discriminating data sources. The system can determine what information would be most useful by analyzing the model. This data can then be cached locally and updated as the sources change to provide more efficient access to the required data.

Second, the use of a discriminating query could result in a different answer from the one produced without the use of the discrimination. This would only happen if the data used for the discrimination is inconsistent with the original data. This is a general problem when integrating heterogeneous information sources. There are two ways to address this problem. First, any system that can retrieve data from multiple sources needs to make it clear where any given data actually came from so the user can assess the accuracy of the data. Second, if the integrity of the final answers is critical, then the system can perform off-line processing to verify that redundant sources contain consistent sets of data.

8 Conclusion

This paper provides an important capability for using sensing actions in query planning to efficiently locate information in a setting that involves a large number of sources. Such optimizations are critical in query planners for information gathering agents since the number of sources available can be quite large. We have argued for the need for exploiting information obtained at run-time and presented an approach that extends classical query planning to exploit run-time information through the use of sensing actions. We also presented an algorithm for building a discrimination matrix, which is used to determine which sensing actions can be used to optimize a plan, and described how existing query planners can exploit the discrimination matrix to insert sensing actions. In future work, we plan to explore approaches to automatically generate rich models of sources by analyzing the contents of sources, which will improve the likelihood of finding useful discriminating queries.

References

[Ambros-Ingerson, 1987] Jose Ambros-Ingerson. *IPEM: Integrated Planning, Execution, and Monitoring.* PhD thesis, Department of Computer Science, University of Essex, 1987.

[Arens *et al.*, 1996] Yigal Arens, Craig A. Knoblock, and Wei-Min Shen. Query reformulation for dynamic information integration. *Journal of Intelligent Information Systems, Special Issue on Intelligent Information Integration*, 6(2/3):99–130, 1996.

[Borgida and Patel-Schneider, 1994] Alexander Borgida and Peter Patel-Schneider. A semantics and complete algorithm for subsumption in the CLASSIC description logic. *Journal of Artificial Intelligence Research*, 1:277–308, 1994.

[Chakravarthy *et al.*, 1990] Upen S. Chakravarthy, John Grant, and Jack Minker. Logic-based approach to semantic query optimization. *ACM Transactions on Database Systems*, 15(2):162–207, 1990.

[Etzioni and Weld, 1994] Oren Etzioni and Daniel S. Weld. A softbot-based interface to the Internet. *Communications of the ACM*, 37(7), 1994.

[Etzioni *et al.*, 1994] Oren Etzioni, Keith Golden, and Dan Weld. Tractable closed-world reasoning with updates. In *Fourth International Conference on Principles of Knowledge Representation and Reasoning*, Bonn, Germany, 1994.

[King, 1981] Jonathan Jay King. *Query Optimization by Semantic Reasoning.* PhD thesis, Stanford University, Department of Computer Science, 1981.

[Knoblock and Levy, 1995] Craig A. Knoblock and Alon Levy, editors. *Information Gathering from Heterogeneous, Distributed Environments*, Technical Report SS-95-08, AAAI Press, Menlo Park, CA, 1995.

[Knoblock *et al.*, 1994] Craig A. 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.

[Knoblock, 1995] Craig A. Knoblock. Planning, executing, sensing, and replanning for information gathering. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, Montreal, Canada, 1995.

[Knoblock, 1996] Craig A. Knoblock. Building a planner for information gathering: A report from the trenches. In *Proceedings of the Third International Conference on Artificial Intelligence Planning Systems*, Edinburgh, Scotland, 1996.

[Kwok and Weld, 1996] Chung T. Kwok and Daniel S. Weld. Planning to gather information. Technical Report UW-CSE-96-01-04, Department of Computer Science and Engineering, University of Washington, 1996.

[Levy *et al.*, 1996] Alon Y. Levy, Anand Rajaraman, and Joann J. Ordille. Query-answering algorithms for information agents. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, Portland, OR, 1996.

[Olawsky and Gini, 1990] Duane Olawsky and Maria Gini. Deferred planning and sensor use. In *Proceedings of the Workshop on Innovative Approaches to Planning, Scheduling and Control*, pages 166–174, San Diego, CA, 1990.