Flexible and Scalable Query Planning in Distributed and Heterogeneous Environments*

José Luis Ambite & Craig A. Knoblock
Information Sciences Institute and Department of Computer Science
University of Southern California
4676 Admiralty Way, Marina del Rey, CA 90292, USA
{ambite, knoblock}@isi.edu

Abstract

We present the application of the Planning by Rewriting (PbR) framework to query planning in distributed and heterogeneous environments. PbR is a new paradigm for efficient high-quality planning that exploits plan rewriting rules and efficient local search techniques to transform an easy-to-generate, but possibly suboptimal, initial plan into a high-quality plan. The resulting planner is scalable, flexible, has anytime behavior, and, applied to query planning, yields a novel combination of traditional query optimization with heterogeneous information source selection. Query planners are the core component of mediator systems, which are becoming increasingly important in a world of interconnected information, and constitute excellent testbeds for planning technology.

Introduction

Query planning is a problem of considerable practical significance. It lies at the core of mediators, systems that integrate information from multiple distributed and heterogeneous sources, and traditional database systems. Mediators are becoming increasingly important given the current explosion of information accessible through networks.

Query planning in mediators presents particular challenges for planning technology. First, it is a highly combinatorial problem, where complex queries have to be composed from the relevant sources among hundreds of available information sources. Second, query plans often have to be produced rapidly. Third, finding any valid plan is not enough, plan quality is also critical. Finally, mediators need to incorporate traditional techniques for query planning in databases and extend them with new capabilities, such as replanning after failures and information gathering actions.

The Planning by Rewriting (Ambite & Knoblock 1997) paradigm is designed to address planning efficiency and plan quality, while providing the benefits of domain-independence. Its characteristics make it especially well-suited for query planning. First, PbR provides a *declarative domain-independent* framework that is easier to understand, maintain and extend than traditional query optimizers. Different query planning domains can be conveniently specified, for example, for different data models such as relational and object-oriented. The uniform specification of the planner facilitates its *extension* with new capabilities, such as learning mechanisms or interleaving planning and execution. Moreover, a general planning architecture fosters *reuse* in the domain specifications and the search methods. For example, the specification of the join operator translates straightforwardly from a relational to an object-oriented model. Likewise, search methods can be implemented once for the general planner and the most appropriate configuration chosen for each particular domain.

Second, PbR scales better than other domain-independent planning algorithms. *Scalability* is critical because of the complexity of query planning in mediators. Third, an important advantage of PbR is its anytime nature, which allows it to trade off planning effort and plan quality. For example, a typical quality metric in query planning is the plan execution time. It may not make sense to keep planning if the cost of the current plan is small enough, even if a cheaper one could be found. Finally, the generality of the PbR framework has allowed the design of a novel combination of traditional query optimization and source selection. In previous work these two types of query processing had been performed in different stages. Because our planner integrates both optimizations in the same search space, it can apply local search techniques and support anytime behavior.

The application of PbR to query planning in mediators, with the resulting integration of traditional query optimization and source selection, is the main contribution of this paper. The remainder of the paper is structured as follows. First, we present the problem of query planning in distributed and heterogeneous environments, and we briefly review the Planning by Rewriting paradigm. Then we describe in detail how PbR is applied to query planning in mediators and show scalability results for several planner configura-

tions. Finally, we discuss related work, future work and conclusions.

Query Planning in Mediators

Mediators provide access to information in distributed and heterogeneous environments. Query planning in mediators involves generating a plan that efficiently computes a user query from the relevant information sources. This plan is composed of data retrieval actions at diverse information sources and operations on this data (such as those of the relational algebra: join, selection, etc.). For an efficient execution, the plan has to specify, first, which data processing operations are going to be needed and in what order, and, second, from which sources each different piece of information should be obtained. The first problem has been the focus of traditional query optimization in databases. The second problem, source selection, is characteristic of distributed and heterogeneous systems. The highly combinatorial nature of query planning in mediators arises from these two independent sources of complexity, namely, the ordering of data processing operators and the selection of relevant information sources for terms in a given query.

Mediators need to provide mechanisms to resolve the semantic heterogeneity among the different sources. Our approach follows that of the SIMS mediator system (Arens, Knoblock, & Shen 1996; Knoblock & Ambite 1997). Briefly, SIMS assumes that a set of information sources such as databases, knowledge bases, web servers, etc., supply data about a particular application domain. The system designer specifies a global model of the application domain and defines the contents of the sources in terms in this global model. A SIMS mediator integrates and provides a single point of access for all the information in such a domain. The user interacts directly with the SIMS mediator expressing queries against the domain model, without knowledge about the schemas or locations of the sources. The global model and the user queries are specified in the Loom description logic (MacGregor 1988), which is the knowledge representation subsystem of the SIMS mediator. Selecting the sources, translating between global domain terms and source terms, and ordering the operations is the task of the query planner.[1]

The specification of the operators for distributed query processing and the encoding of information goals is presented in (Knoblock 1996). A sample operator, `join`, is shown in Figure 1. The join operator takes two subqueries, that are available locally at the mediator and combines them using some conditions to produce the joined query. Other operators include retrieve, selection, assignment, and union. A sample information goal is shown in Figure 2. This goal asks to send to the output device of the mediator all the names of airports in Tunisia. Figure 3 shows two plans, of different quality, that compute this query by joining data from different sources. Query processing in mediators is further described after we review the general Planning-by-Rewriting framework.

[1] We explain how the knowledge representation system and the query planner interact in the section on the application of PbR to query planning in mediators.

```
(define (operator join)
  :parameters (?query ?jconds ?query-a ?query-b)
  :precondition
   (:and
     (available local ?query-a)
     (available local ?query-b)
     (join-query ?query ?jconds ?query-a ?query-b))
  :effect (available local ?query))
```

Figure 1: Sample Operator

```
(available output
  (sims-retrieve (?ap_name)
    (:and (airport ?aport)
          (country-name ?aport "Tunisia")
          (port-name ?aport ?ap_name))))
```

Figure 2: Sample Information Goal

Review of Planning by Rewriting

Planning by Rewriting (Ambite & Knoblock 1997) follows the iterative improvement style of many optimization algorithms. The framework works in two phases:

1. Efficiently generate an initial solution plan.
2. Iteratively rewrite the current solution plan in order to improve its quality using a set of declarative plan rewriting rules until either an acceptable solution is found or a resource limit is reached.

In Planning by Rewriting a plan is represented by a graph notation in the spirit of partial-order causal-link planners such as UCPOP (Penberthy & Weld 1992). The nodes are domain actions. The edges specify a temporal ordering relation among nodes, imposed by causal links and ordering constraints.

A plan rewriting rule, akin to term and graph rewriting rules, specifies the replacement under certain conditions of a partial plan by another partial plan. Our system ensures that the rewritten plan remains complete and consistent. These rules are intended to improve the quality of the plans. Figures 5, 7, and 8 in the next section are examples of plan rewriting rules in the query planning domain.

The following is a list of the main issues in Planning by Rewriting. A detailed description of the general Planning by Rewriting paradigm is given in (Ambite & Knoblock 1997). The next section discusses these issues in the context of query planning.

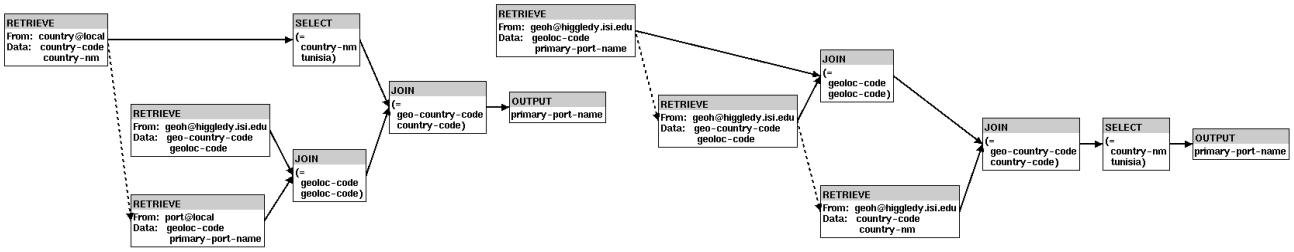

Figure 3: Sample Query Plans: Optimal (left) and Suboptimal (right)

- *Efficient generation of an initial solution plan.* In many domains obtaining a possibly suboptimal initial plan is easy.
- *Definition and application of the plan rewriting rules.* The user can specify appropriate rewriting rules for a domain in a simple, but quite general, rule definition language. These rules are matched against the current plan and generate new transformed plans of possibly better quality.
- *Plan quality measure.* This is the plan cost function of the application domain that should be optimized during the planning process.
- *Search of the space of rewritings.* There are many possible ways of searching the space of rewritten plans, for example, gradient descent, simulated annealing, etc.

Application of Planning by Rewriting to Query Planning in Mediators

For the Planning by Rewriting framework to be applicable, there must exist an efficient mechanism to generate an initial solution plan. It is desirable that this mechanism also be able to produce several (possibly random) initial plans on demand. Both properties are satisfied by the query planning domain. Initial query evaluation plans can be efficiently obtained as random depth-first search parses of the query. These initial plans although correct may be of very low quality.

For our query planning domain the quality of a plan is its execution cost. The execution cost of a distributed query plan depends on the size of intermediate results, the cost of performing data manipulation operations (e.g., join, selection, etc.), and the transmission through the network of the intermediate results from the remote sources to the mediator. We estimate the execution cost based on the expected size of the intermediate results. We assume that the transmission and processing costs are proportional to the size of the data involved. The query size estimation is computed from simple statistics obtained from the source relations, such as number of tuples in a relation, the number of distinct values for each attribute, and the maximum and minimum values for numeric attributes.

A randomly generated initial plan for the query in Figure 2 is shown on the right side of Figure 3. This plan has a much lower quality than the optimal plan (shown on the left in Figure 3). Note the three sequential retrievals from the same source, `geoh`, at the same host, `higgledy.isi.edu`, not taking advantage of the possibility of executing the queries in parallel at different hosts. Also it is generally more efficient to perform selections as early as possible in order to reduce the data that the subsequent steps of the plan must process. This initial plan performs the selection step last as opposed to the plan on the left in Figure 3 where it is done immediately after the corresponding retrieve.

The core of the planning process consists of the iterative application of a set of plan rewriting rules until a plan of acceptable quality is found. In our query planning domain, the rules can be grouped into three classes. The first class of rules is derived from the properties of the distributed environment. A logical description of these rules is shown in Figure 4. The `Source-Swap` rule allows the planner to explore the choice of alternative information sources that can satisfy the same query but may have different retrieval or transmission costs. This rule is not only necessary for query plan optimization but it also serves as a repair rule when planning and execution are interleaved. Suppose that the planner started executing a plan and one of the sources needed went down, then the subquery sent to that source will fail. By applying this rule PbR can repair the plan and complete the execution without replanning and re-executing from scratch. The other three rules rely on the fact that, whenever possible, it is generally more efficient to execute a group of operators together at a remote information source than to transmit the data over the network and execute the operations at the local system. Note the need for checking the capabilities of the information sources. We do not assume that sources are full databases. They may have no query processing capabilities (for example, wrapped WWW pages) or support very limited types of queries (for example WWW forms). Figure 5 shows the `Remote-Join-Eval` rule in the input syntax accepted by the PbR planner. This rule specifies that if in a plan there exist two retrieve operators from the

same remote database which are consequently joined, and the remote source is capable of performing joins, the system can rewrite the plan into one that contains a single retrieve operation that pushes the join operation to the remote database.

Source-Swap:
$retrieve(Q, Source1) \quad \wedge$
$alternative\text{-}source(Q, Source1, Source2)$
$\Rightarrow retrieve(Q, Source2)$

Remote-Join-Eval:
$(retrieve(Q1, Source) \bowtie retrieve(Q2, Source))$
$\wedge capability(Source, join)$
$\Rightarrow retrieve(Q1 \bowtie Q2, Source)$

Remote-Selection-Eval:
$\sigma_A retrieve(Q1, Source) \wedge capability(Source, selection)$
$\Rightarrow retrieve(\sigma_A Q1, Source)$

Remote-Assignment-Eval:
$assign_{X:=f(Ai)} retrieve(Q1(Ai), Source) \quad \wedge$
$capability(Source, assignment)$
$\Rightarrow retrieve(assign_{X:=f(Ai)} Q1(Ai), Source)$

Figure 4: Transformations (Distributed Environment)

```
(define-rule :name remote-join-eval
  :if (:operators
       ((?n1 (retrieve ?query1 ?source))
        (?n2 (retrieve ?query2 ?source))
        (?n3 (join ?query ?jc ?query1 ?query2)))
       :constraints ((capability ?source 'join)))
  :replace (:operators (?n1 ?n2 ?n3))
  :with (:operators
         ((?n4 (retrieve ?query ?source))))
```

Figure 5: Remote-Join-Eval Rewriting Rule

The second class of rules are derived from the commutative, associative, and distributive properties of the operators of the relational algebra. A logical description of these rules is shown in Figure 6. The `Join-Swap` rule in the PbR syntax is shown in Figure 7. This rule specifies that two consecutive joins operators can be reordered and allows the planner to explore the space of join trees. In our query planning domain (Knoblock 1996) queries are expressed as complex terms. The PbR rules use the interpreted predicates in the `constraints` field to manipulate such query expressions. For example, the `join-swappable` predicate checks if the two join operators have queries that can be exchanged. This user-defined predicate takes as input the description of the two join operations (the first eight variables) and produces as output the description of the two reordered join operations (as bindings for the last eight variables). If two subqueries do not share any attributes, the join degenerates into a cross-product. Although a cross-product is inefficient, such rewritings are allowed for completeness.

The third set of rewriting rules arises from the heterogeneous nature of the environment. As we explained earlier, a mediator needs to reconcile the semantic differences among the sources. A common approach (Arens, Knoblock, & Shen 1996; Levy, Rajaraman, & Ordille 1996; Kwok & Weld 1996) is to define each source class as a logical formula over classes in a global domain model. In the SIMS mediator, for each class and set of attributes in the domain model the system automatically compiles axioms that describe how to obtain such information by combining a set of sources. A set of maximal axioms is precompiled when the domain model is defined. The set of relevant axioms for a given user query can be efficiently computed by instantiating these maximal axioms at run time. A detailed explanation of this process lies outside the scope of this paper, see (Ambite *et al.* 1998). For the purposes of this paper we will consider these axioms as given. Our system automatically derives query-specific plan rewriting rules from these integration axioms in order to explore the alternative ways of obtaining each class of information in a user query. A sample axiom for the query in Figure 2 is:

```
airport(country-code port-name) ⇔
  airport(geoloc-code port-name) ∧
  location(geoloc-code country-code)
```

This axiom states that in order to obtain the attributes `country-code` and `port-name` of the `airport` class, the system needs to join data from two sources. The first source provides `geoloc-code` and `port-name` of `airports`, and the second provides `geoloc-code`

Join-Swap:
$Q1 \bowtie (Q2 \bowtie Q3) \Leftrightarrow Q2 \bowtie (Q1 \bowtie Q3) \Leftrightarrow Q3 \bowtie (Q2 \bowtie Q1)$

Selection-Swap: $\sigma_A(Q1 \bowtie Q2) \Leftrightarrow \sigma_A Q1 \bowtie Q2$

Assignment-Swap: $assign_{X:=f(Ai)}(Q1(Ai) \bowtie Q2) \Leftrightarrow assign_{X:=f(Ai)} Q1(Ai) \bowtie Q2$

Join-Union-Distribution:
$Q1 \bowtie (Q2 \cup Q3) \Leftrightarrow (Q1 \bowtie Q2) \cup (Q1 \bowtie Q3)$

Figure 6: Transformations (Relational Algebra)

```
(define-rule :name join-swap
  :if (:operators
       ((?n1 (join ?q1 ?jc1 ?sq1a ?sq1b))
        (?n2 (join ?q2 ?jc2 ?sq2a ?sq2b)))
       :links (?n1 ?n2)
       :constraints
       (join-swappable ?q1 ?jc1 ?sq1a ?sq1b  ;in
                       ?q2 ?jc2 ?sq2a ?sq2b  ;in
                       ?q3 ?jc3 ?sq3a ?sq3b  ;out
                       ?q4 ?jc4 ?sq4a ?sq4b));out
  :replace (:operators (?n1 ?n2))
  :with (:operators
         ((?n3 (join ?q3 ?jc3 ?sq3a ?sq3b))
          (?n4 (join ?q4 ?jc4 ?sq4a ?sq4b)))))
```

Figure 7: Join-Swap Rewriting Rule

and country-code of geographic locations. In our model the class airport is a subclass of location so it inherits its attributes. Thus, the airport class has three attributes: geoloc-code, country-code, and port-name. The corresponding rewriting rule is shown in Figure 8. This rule states that if in a plan there is a set of steps (?nodes) that obtain attributes country-code and port-name of the airport class, the planner could alternatively obtain this information using the axiom shown above. That is, the ?nodes identified in the rule antecedent will be removed from the plan and replaced by the join and the two retrieve steps in the rule consequent. This type of rewriting rules resemble task expansion in Hierarchical Task-Network Planning (Erol, Nau, & Hendler 1994; Tate 1977), although in PbR they are used for local search instead of generative planning.

```
(define-rule :name
  (<=> (airport country-code port-name)
       (:and (airport geoloc-code port-name)
             (location geoloc-code country-code)))
 :if (:constraints
      ((identify-axiom-steps
        (airport country-code port-name) ?nodes)))
 :replace (:operators ?nodes)
 :with
 (:operators
   ((?n1 (retrieve port@local
           (airport geoloc-code port-name)))
    (?n2 (retrieve geoh@higgledy.isi.edu
           (location geoloc-code country-code)))
    (?n3 (join (airport country-code port-name)
           ((= geoloc-code.1 geoloc-code.2))
           (airport geoloc-code.1 port-name)
           (location geoloc-code.2 country-code))
))))
```

Figure 8: Rewriting Rule for Integration Axiom

The space of rewritings for query planning is too large for complete search methods to provide an acceptable performance. The fact that in many cases, such as query planning, the quality of a plan can only be estimated supports the argument for possibly incomplete search strategies, such as gradient descent. The effort spent in finding the global optimum may not be justified given that the cost function only captures approximately the real costs in the domain. As the accuracy of the cost model increases the planner may perform a more complete search of the plan space. In these cases simulated annealing may be more appropriate than strict gradient descent. In order to explore the space of query plans our planner currently uses variations of gradient descent (steepest and first-improvement) with random restart to escape low-quality local minima and a fixed-length random walk to traverse plateaus.

Results in Query planning

We performed two experiments to test the scalability of the PbR approach applied to query planning. In the first experiment, we compare the behavior of PbR and Sage in a distributed query planning domain as the size of the queries increases. In the second experiment, we compare Sage and PbR in a distributed and heterogeneous domain by increasing the number of alternative sources per domain class.

For the first experiment we generated a synthetic domain for the SIMS mediator and defined a set of conjunctive queries involving from 1 to 20 domain classes. The queries have one selection on each class. Each source contains two classes and can perform remote operations. We present results for three planners:

Sage: This is the original query planner (Knoblock 1995; 1996) for the SIMS mediator, which performs a best-first search with a heuristic commonly used in query optimization that explores only the space of left join trees. It generates optimal left-tree query plans.

Initial: This is the initial plan generator for PbR. It generates random depth-first search parses of the query. It is the fastest planner but may produce very low quality plans.

PbR: We used the Join-Swap, Remote-Selection--Eval, Remote-Join-Eval, and Source-Swap rewriting rules introduced above. The search strategy is first-improvement gradient descent and it picks the best plan after three random restarts.

The results of the first experiment are shown in Figures 9 and 10. Figure 9 shows the planning time in a logarithmic scale. The times for PbR include both the generation of the three random initial plans and their rewriting. The times for Initial are the average of the three random depth-first parses of each query. Sage is able to solve queries involving up to 8 classes, but larger queries cannot be solved within the search limit of 200,000 nodes. PbR scales better and solved all queries with low cost plans. Figure 10 shows the quality of the query plans for the three planners. A logarithmic scale is used because of the large absolute values of the plan costs. PbR rewrites the very poor quality plans generated by Initial into high-quality plans. The quality of the plans produced by PbR and Sage is comparable for the range tractable for Sage (in fact PbR produces better quality plans because it searches the larger space of bushy query trees) and beyond that range it scales gracefully.

In the second experiment, we test the scalability of PbR as the number of alternative sources for a class of information in the domain increases. We defined a set of synthetic domains with only five domain classes but increasing the number of sources per domain class from 1 to 100. In Figure 11 we show the planning time for PbR and Sage for queries involving four and five classes. Sage can solve the five-class queries in domains

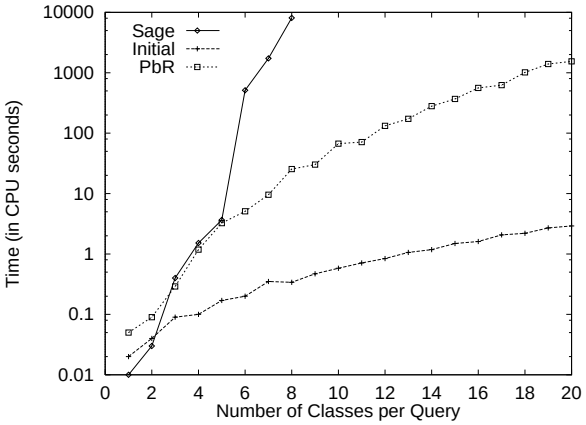

Figure 9: Time: Distributed

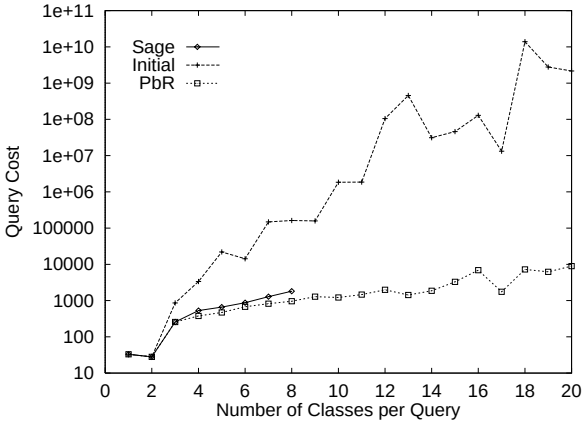

Figure 10: Quality: Distributed

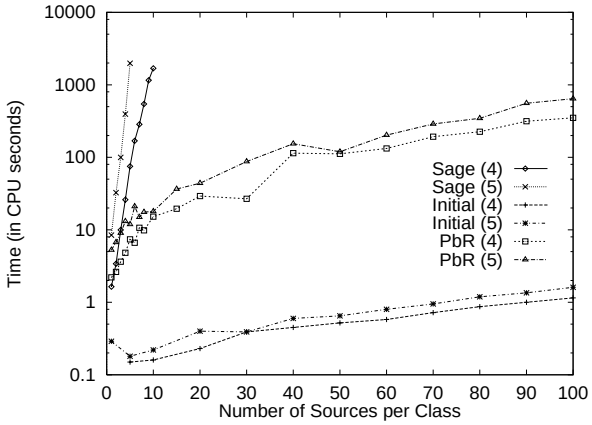

Figure 11: Time: Distributed and Heterogeneous

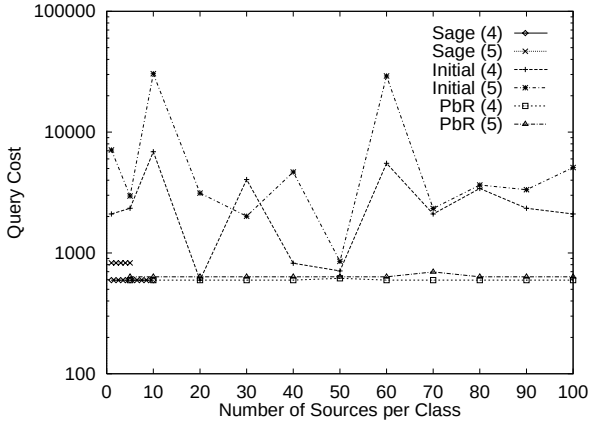

Figure 12: Quality: Distributed and Heterogeneous

up to five sources per class and the four-class queries up to ten sources per class within its 200000 nodes search limit. PbR scales up much better solving the queries for all the domains. In order to control for quality we assigned the same cost of access to all sources. Thus we know that the cost of the optimal plan is constant. Figure 12 shows that PbR closely approximates the optimal cost regardless of the increasing complexity of the search space and the very high cost of many initial plans.

Related Work

In the database literature, query optimization has been extensively studied (Jarke & Koch 1984). Query optimizers attempt to both find the most efficient algebraic form of a query and to choose specific methods to implement each data processing operation (Graefe 1993). For example, a join can be performed by a variety of algorithms, such as nested loops, merge scan, hash join, etc. In the present paper we have focused on the algebraic part of query optimization because in our distributed environment the mediator does not have any control over the optimizations employed in remote databases, and, so far, the size of data the mediator needs to manipulate locally has not required very sophisticated consideration of implementation algorithms.

The research on query optimization most relevant to our approach lies in three areas. The first area is distributed query optimization. An algorithm for distributed query optimization based on query tree transformation is presented in (Chu & Hurley 1982). Our relational algebra plan rewriting rules are similar to their transformations, but our system accepts arbitrary specification of rules as opposed to a hand-coded algorithm. The second area is declarative and extensible query optimizers. An extensible query optimizer based on query rewriting was implemented for the Startbust system (Pirahesh, Hellerstein, & Hasan 1992). They also define declarative rules to manipulates graphs, but they rely heavily in procedural attachments to implement the preconditions and effects of rules. They handle full SQL queries on centralized databases, but they do not deal with semantic heterogeneity. Our

rules currently do not cover the aggregation operators of SQL, although we could incorporate rewrites similar to those in (Yan & Larson 1995). Another influential work in query optimization is Exodus (Graefe & DeWitt 1987). Exodus is a query optimizer generator that compiles a query optimizer out of a given set of operators, transformation rules and the code for the methods that implement each operator. Although Exodus strives for extensibility, its operator definition language is more restricted than ours. Also it has a fixed search strategy (a form of hill climbing). Exodus focuses more in implementation methods for relational operators and it operates on centralized databases. Volcano (Graefe *et al.* 1994), a successor of Exodus, provides a general implementation of data processing operations based on iterators, but does not offer more generality on algebraic query optimization. Interestingly, it provides a simple form of contingency planning in which two alternative implementation methods are included in the plan and which one is chosen depends on the value of a parameter at execution time. We expect that PbR would be able to provide more general interleaving of planning and execution (Knoblock 1995). The third area of work is on efficient search algorithms for query planners (Swami 1989; Ioannidis & Kang 1990). Since our approach is based on a domain-independent planner it is more flexible than previous research, supporting the integration of different domains (operators and rewriting rules) and search algorithms in an uniform and easily extensible framework.

Despite the practical importance of query planning, there has been little work in the planning literature. Occam (Kwok & Weld 1996) is a planner for information gathering that focuses on the source selection problem. Our work combines both source selection and traditional query optimization. Sage (Knoblock 1996) considers plan quality and supports interleaving of planning and execution. PbR does not currently interleave planning and execution, but it is as general as Sage with better scaling properties as shown in the results section.

The framework of Planning by Rewriting is related to several pieces of previous work in AI planning. Most significantly it is a generalization of plan merging (Foulser, Li, & Yang 1992) and it follows on iterative repair ideas such as those in (Minton 1992) and (Zweben, Daun, & Deale 1994). A more detailed discussion appears in (Ambite & Knoblock 1997).

Finally, PbR can be understood as an instantiation of the local search idea, which has a long tradition in combinatorial optimization (Papadimitriou & Steiglitz 1982). However, instead of hard-coding a specific algorithm for each problem, PbR provides a general framework in which a problem is cast as a declarative planning specification and a set of declarative rewriting rules, while the bulk of the program consisting of the rewriting and search engine is reused.

Conclusions and Future Work

We have presented the application of the Planning by Rewriting framework to the challenging domain of query planning in mediators. As a result we have developed a novel combination of traditional query optimization and source selection. PbR explores this integrated optimization space using efficient local search methods that make the planner scalable to large queries and domains with large number of alternative sources.

Query planning is an excellent domain to test planning techniques, and, in particular our Planning by Rewriting framework. We plan to extend the capabilities of PbR in several dimensions: more sophisticated query planning domains, interleaving of planning and execution, insertion of information gathering actions, and new search methods. More complex query planning domains will serve both to test the expressiveness of our rule definition language and to compare with query optimizers in the database literature. Interleaving planning and execution is necessary in order to deal effectively with unexpected situations in the environment such as database or network failures. It also enables the planner to perform dynamic query optimization in which plans depend on run-time conditions, and to insert information gathering actions (Ashish, Knoblock, & Levy 1997).

We plan to explore a variety of search techniques for query planning, for example, variable depth rewriting. In variable depth search a sequence of rewritings is applied atomically. This allows the planner to overcome initial cost increases that eventually would lead to strong cost reductions. This idea leads to the creation of rule programs which is particularly appealing in the query planning domain in which sequences of rewritings are natural. For example, a sequence of `Join-Swap` transformations may put two retrieve operators on the same database together in the query tree and then `Remote-Join-Eval` would collapse the explicit join operator and the two retrieves into a single retrieval of a remote join.

Acknowledgments

This work was supported in part by the United States Air Force under contract number F49620-98-1-0046, by the Rome Laboratory of the Air Force Systems Command and the Defense Advanced Research Projects Agency (DARPA) under contract number F30602-97-2-0352, and by a research grant from General Dynamics Information Systems. The views and conclusions contained in this paper are the authors' and should not be interpreted as representing the official opinion or policy of any of the above organizations or any person connected with them. The authors wish to thank the anonymous reviewers for their detailed and useful comments.

References

Ambite, J. L., and Knoblock, C. A. 1997. Planning by rewriting: Efficiently generating high-quality plans. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*.

Ambite, J. L.; Knoblock, C. A.; Muslea, I.; and Philpot, A. 1998. Compiling source descriptions for efficient and flexible information integration. Submitted.

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

Ashish, N.; Knoblock, C. A.; and Levy, A. 1997. Information gathering plans with sensing actions. In *Proceedings of the Fourth European Conference on Planning*.

Chu, W. W., and Hurley, P. 1982. Optimal query processing for distributed database systems. *IEEE Transactions on Computers* 31(9):835–850.

Erol, K.; Nau, D.; and Hendler, J. 1994. UMCP: A sound and complete planning procedure for hierarchical task-network planning. In *Proceedings of the Second International Conference on Artificial Intelligence Planning Systems*, 249–254.

Foulser, D. E.; Li, M.; and Yang, Q. 1992. Theory and algorithms for plan merging. *Artificial Intelligence* 57(2–3):143–182.

Graefe, G., and DeWitt, D. J. 1987. The EXODUS optimizer generator. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*.

Graefe, G.; Cole, R. L.; Davison, D. L.; McKenna, W. J.; and Wolniewicz, R. H. 1994. Extensible query optimization and parallel execution in volcano. In J. C. Freytag, G. Vossen and D. Maier., ed., *Query Processing for Advanced Database Applications*. San Francisco, California: Morgan Kaufmann. 305–381.

Graefe, G. 1993. Query evaluation techniques for large databases. *ACM Computing Surveys* 25(2):73–170.

Ioannidis, Y., and Kang, Y. C. 1990. Randomized algorithms for optimizing large join queries. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 312–321.

Jarke, M., and Koch, J. 1984. Query optimization in database systems. *ACM Computing Surveys* 16(2):111–152.

Knoblock, C. A., and Ambite, J. L. 1997. Agents for information gathering. In Bradshaw, J., ed., *Software Agents*. Menlo Park, CA: AAAI/MIT Press.

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

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

Kwok, C. T., and Weld, D. S. 1996. Planning to gather information. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*.

Levy, A. Y.; Rajaraman, A.; and Ordille, J. J. 1996. Query-answering algorithms for information agents. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*.

MacGregor, R. 1988. A deductive pattern matcher. In *Proceedings of the Seventh National Conference on Artificial Intelligence*.

Minton, S. 1992. Minimizing conflicts: A heuristic repair method for constraint-satisfaction and scheduling problems. *Artificial Intelligence* 58(1-3):161–205.

Papadimitriou, C. H., and Steiglitz, K. 1982. *Combinatorial Optimization: Algorithms and Complexity*. Englewood Cliffs, NJ: Prentice Hall.

Penberthy, J. S., and Weld, D. S. 1992. UCPOP: A sound, complete, partial order planner for ADL. In *Third International Conference on Principles of Knowledge Representation and Reasoning*, 189–197.

Pirahesh, H.; Hellerstein, J. M.; and Hasan, W. 1992. Extensible/rule based query rewrite optimization in starburst. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*.

Swami, A. N. 1989. Optimization of large join queries: Combining heuristic and combinatorial techniques. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 367–376.

Tate, A. 1977. Generating project networks. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, 888–893.

Yan, W. P., and Larson, P.-A. 1995. Eager aggregation and lazy aggregation. In McLeod, D.; Sacks-Davis, R.; and Schek, H., eds., *Proceedings of 21th International Conference on Very Large Data Bases*.

Zweben, M.; Daun, B.; and Deale, M. 1994. Scheduling and rescheduling with iterative repair. In *Intelligent Scheduling*. San Mateo, CA: Morgan Kaufman. 241–255.