Applying a General-Purpose Planner to the Problem of Query Access Planning*

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

Abstract

Researchers have extensively studied the issues that arise in applying a general-purpose planner to the problem of stacking blocks. The interesting question is whether these are the same issues that arise in solving real-world problems. In this paper we focus on the problem of applying a general-purpose planner to generate query access plans for a multidatabase system. We identify both the representational issues and search issues that arise in applying a planner to this problem. It turns out that to represent this problem requires a representation of complex objects, ability to reason about resource constraints, and a functional interface to handle the low-level details. The search for a plan differs from traditional work in that the quality of the plans is important since there are many possible plans, and the system needs to reliably produce plans in a short amount of time.

Introduction

After many years of work in planning on the blocks world and other toy problems, there is now a great deal of interest in "real-world" planning problems. This paper examines the issues in applying a general-purpose planner to a specific planning application, called query access planning. Query access planning involves building plans for processing data to answer queries. The plans include the source for the information, the specific operations that are to be performed on the data, and the order in which the operations are to be performed. The general problem is quite hard since there is a very large number of ways a query can be processed. The choice of plans is critical since the cost of executing different plans for the same query can range from a few milliseconds to a few years.

The database community has been building specialized systems for performing query access planning for many years (Jarke and Koch 1984, Yu and Chang 1984). In order to address this problem these systems exploit a variety of heuristics and techniques to limit the search space. These approaches have been carefully optimized for the problem of query processing in the single database environment and the distributed database environment. However, there has been less work on the problem of query processing for multidatabase systems (Landers and Rosenberg 1982, Reddy *et al.* 1989) where there are a set of distributed, autonomous, and heterogeneous databases. The existing approaches in this environment are inflexible in the sense that choice of the information sources for a given query is fixed and the integration of this information is predefined. To address this problem we have applied a general-purpose planner to the problem of generating query access plans for multidatabase queries.

The advantage of using a general-purpose planner instead of an specialized algorithm are three-fold. First, the declarative representation of the problem simplifies the problem of augmenting and refining the planning problem. Second, explicit plan representation makes it possible to interleave planning and execution to provide a much more flexible query processor. This interleaving allows the handling of execution failures and the ability to cope with incomplete and inconsistent information. Third, the planning framework allows us to declaratively specify the heuristics for searching the space of query access plans. These heuristics can be hand-crafted or automatically generated using the various speedup learning techniques to improve efficiency and improve the quality of the plans produced.

Previous planning work has looked at the problem of efficiently merging query access plans (Yang *et al.* 1992) and trading off search time versus execution time in optimizing query access plans (Shekhar and Dutta 1989). All of this previous work assumes that the query access plans are given.

This paper focuses on the issues in applying a

*The research reported here was 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 views and conclusions contained in this report are those of the authors and should not be interpreted as representing the official opinion or policy of RL, ARPA, NSF, the U.S. Government, or any person or agency connected with them.

```
(available output sims
  (retrieve (?port_name ?depth ?ship_type ?draft)
            (:and (seaports ?port)
                  (seaports.port_nm ?port ?port_name)
                  (seaports.glc_cd ?port ?glc_cd)
                  (channels ?channel)
                  (channels.glc_cd ?channel ?glc_cd)
                  (channels.ch_depth_ft ?channel ?depth)
                  (notional_ship ?ship)
                  (notional_ship.sht_nm ?ship ?ship_type)
                  (notional_ship.range ?ship ?range)
                  (> ?range 10000)
                  (notional_ship.max_draft ?ship ?draft)
                  (< ?draft ?depth))))
```

Figure 1: Example Planner Goal

general-purpose planner to generate query access plans. First, we describe the representation of the problem and features required of a planner to make this representation possible. Second, we describe the issues in generating a query access plan. Unlike most existing planning domains, the quality of the plan produced and the ability to produce this plan in a limited amount of time are critical issues. Third, we discuss the implications of this work to research on planning and learning.

Representing the Problem

Query access planning requires developing an ordered set of operations for generating a requested set of data. This includes selecting the source for the data, the operations for processing the data, the sites where the operations will be performed and the order in which to perform them. Since data can be moved around between different sites, processed at different locations, and the operations can be performed in a variety of orders, the space of possible plans is quite large. In this section we will describe the representation of the problem in a planning system and in the next section we will describe how the planner is used to solve the problem.

The definition of a planning problem consists of a goal, an initial state, and a set of operators that can be applied to transform the initial state into the goal. For query access planning, the goal of a problem consists of a description of a set of data as well as the information source and server where that data is to be sent. For example, Figure 1 illustrates a goal which specifies that the set of data be sent to the OUTPUT device of the SIMS information server (Arens *et al.* 1993). The data to be retrieved is defined by the query expressed in the Loom knowledge representation language (MacGregor 1990).

This particular query requests all seaports and corresponding ships with a range greater than 10,000 that can be accommodated within each port. The first argument to the **retrieve** expression is the parameter list, which specifies which parameters of the query to return. The second argument is a description of the information to be retrieved. This description is expressed as a conjunction of concept and relation expressions, where the concepts describe classes of information, and the relations describe constraints on these classes. The first subclause of the query is an example of a concept expression and specifies that the variable **?port** ranges over members of the class **seaport**. The second subclause is an example of a relation expression and states that the relation **port_name** holds between the value of **?port** and the variable **?port_name**. The query describes a class of seaports and a class of ships, and requests all seaport and ship pairs where the depth of the port exceeds the draft of the ship.

The initial state of a problem defines the information servers that are available as well as which server provides which information sources. The example shown in Figure 2, defines two servers, one available at ISI and another available at Unisys, where both servers contain the GEO and ASSETS information sources. In addition to this information, there is also some knowledge of the contents of the information sources that is stored in a Loom knowledge base. However, this information is static and is accessed directly via Lisp functions rather than through the initial-state data structure.

```
((server-available isi)
 (server-available unisys)
 (server geo unisys)
 (server assets unisys)
 (server geo isi)
 (server assets isi))
```

Figure 2: Example Initial State

There are five general operators that are used to plan out the processing of a query:

- Move – Moves a set of data from one information source to another information source.
- Join – Combines two sets of data into a combined set of data using the given join relation.

```
(define (operator join)
    :parameters (?join-ops ?data ?data-a ?data-b)
    :precondition (:and (join-partition ?data ?join-ops
                                        ?data-a ?data-b)
                        (available local sims ?data-a)
                        (available local sims ?data-b))
    :effect (:and (available local sims ?data)
                  (:not (available local sims ?data-a))
                  (:not (available local sims ?data-b))))
```

Figure 3: The Join Operator

```
(define (operator move-data)
    :parameters (?db1 ?server1 ?db2 ?query)
    :resources ((resource ?server1 server))
    :precondition (:and (available ?db1 ?server1 ?query)
                        (:neq ?db1 ?db2)
                        (:neq ?db1 output)
                        (:or (:eq ?db2 local)
                             (:eq ?db2 output)))
    :effect (:and (:not (available ?db1 ?server1 ?query))
                  (available ?db2 sims ?query)))
```

Figure 4: The Move Operator

- Retrieve – Specifies the data that is to be retrieved from a particular information source.
- Select – Selects a subset of the data using the given constraints.
- Compute – Constructs a new term in the data from some combination of the existing data.

Each of these operations manipulates one or more sets of data, where the data is specified in the same terms that are used for specifying the original query.

Consider the operator shown in Figure 3 that defines a join performed in the local system. This operator is used to achieve the goal of making some information available in the local knowledge base of the SIMS server. It does this by partitioning the request into two subsets of the requested data, getting that information into the local system and then joining that data together to produce the requested set of data. The operator states that (1) if a query can be broken down into two subqueries that can be joined together over some join relation, and (2) the first set of data can be made availabe in the local system, and (3) the second set of data can be made available in the local system, then the requested information can be made available. As a side effect, the intermediate data will no longer be available. The predicate **join-partition** is defined by a piece of Lisp code that produces the possible join partitions of the requested data.

Figure 4 shows the **move-data** operator, which moves data from one information source to another. The plans generated by the system attempt to exploit the parallelism available by issuing subqueries in parallel (Knoblock 1994). In order to avoid resource conflicts where two queries are sent to the same server at the same time, the resources are explicitly represented in the resources field. In the move operator shown in Figure 4, the server used in a particular query is declared as a resource.

Beyond the standard functionality provided by most general-purpose planning systems, there are three capabilities of a planner required to represent this problem: the ability to represent and manipulate complex terms, the ability to define preconditions as functions, and the explicit representation of resources. The representation of complex terms is essential since the specification of a set of data can be quite involved. It includes the specification of the classes of objects, the relevant relations on those objects, constraints on the individual objects as well as constraints between objects, and so on. An example of a complex term is shown in the example goal of Figure 1. Terms such as these would be impractical to define as a flat literal with a fixed number of parameters. It would also be difficult to plan for if it were represented by a set of flat literals since it would require reasoning about the achievement of several simultaneous goals and existing planners are not particularly well suited to that problem.

The functional interface is important for representing this problem at an appropriate level of abstraction. The aspect of this problem that is well suited to a planner is the search through the space of possible operations for manipulating the data. There are other aspects to this problem such as computing the possible partitions of a query or determining whether a set of data is contained in a particular information source that would be difficult to capture in a planner. If these types of queries could be anticipated, then they could be explicitly encoded in the initial state. However, that would be completely impractical for a general query-

answering system since there are a potentially infinite set of possible queries.

Finally, the explicit representation of resources is required to generate parallel query access plans. The use of parallel plans is quite useful in reducing the overall execution time of the queries. The representation of the resources allows the system to explicitly reason about potential resource conflicts and take them into account in order to generate the best plan to solve the problem. This problem could also be addressed by scheduling the operations after the planning is complete; however, this approach would be less efficient since there may be several different operators for achieving the same goal and the choice of operator could have a significant impact on the schedule.

Generating a Query Access Plan

The generation of a query access plan is a planning problem, but has two important aspects that differentiates it from other planning problems. First, queries must be answered in real time and so the plans must be constructed in real time. Second, the problem is not to find a solution, but rather to find a good solution. This problem could always be solved by bringing all of the information into the local system, processing it locally, and returning the result. However, this approach could require years of processing for some queries that could be solved in other ways in a matter of seconds.

The planner is implemented in a version of UCPOP (Barrett *et al.* 1993) that has been modified to generate parallel execution plans (Knoblock 1994). The system currently searches through the space of possible plans using a best-first search until a complete plan is found. This approach to searching the space is only possible on moderate-size queries that involve three to four joins. We are currently exploring more intelligent search methods that will allow the system to handle larger queries.

The plan generated for the example query in Figure 1 is shown in Figure 5. In this example, the system partitions the given query such that the ship information is retrieved in a single query to the ASSETS information source and the seaport and channel information is retrieved in a single query to the GEO information source. All of the information is brought into the local system (Loom) where the draft of the ships can be compared against the depth of the seaports. Once the final set of data has been generated, it is returned by the system.

The system searches for a plan that can be implemented as efficiently as possible. To do this the planner must take into account the cost of accessing the different information sources, the cost of retrieving intermediate results, and the cost of combining these intermediate results to produce the final results. In addition, since the information sources are distributed over different machines or even different sites, the planner takes advantage of potential parallelism and generates subqueries that can be issued concurrently.

To search the space of query access plans efficiently, the system uses a simple estimation function to calculate the expected cost of the various operations. Using this evaluation function in a best-first search, the system will eventually produce a plan that has the lowest overall parallel execution cost. In the example, the planner leaves the join between the seaports and channels to be performed by the remote GEO information server since this will be cheaper than moving the information into the local system. If the system could perform all of the work in one remote system, then it would completely bypass the local system and send the data directly to the output.

We are in the process of investigating different approaches to finding the best plan available in the time allowed. This is potentially a good application for learning, but differs from other types of domains where learning systems have been applied. Since there are many possible solutions to any given problem, what needs to be learned is a set of heuristics that will quickly produce a set of high-quality solutions.

Discussion

This abstract described a particular planning application and the issues that arose in casting this problem as a general planning problem. First, there are the issues of representing the problem. The three critical features that are not typically provided by planners are the representation of complex terms, a functional interface, and the explicit representation of resources. Second, there are issues that arise in generating plans in this domain that differ from more traditional domains. In particular, the quality of the solution is a critical aspect in solving this problem and the system need to generate a solution in a limited amount of time.

In looking at the issues that arise in real-world planning problems, I would claim that much of the existing work on planning has focused on the wrong problems. Research on planning has spent a great deal of effort handling more expressive representations and handling interactions between operators. While these are important problems, other issues such as efficiently searching the space of plans and generating high qualify plans have been neglected. Most of the work on speed-up learning for planning does not fully address this problem since these systems often reduce an exponential search to a smaller exponential search, but they do not reduce search to the point where realistic problems in these domains are tractable. As evidence of this fact note that the planners that have been applied to real problems (Wilkins 1988, Currie and Tate 1991) do not use any of the learning techniques, but rather have carefully engineered domains that allow problems to be solved with very little search. When people build specialized planners for specific applications they develop techniques and heuristics that make some interesting class of problems

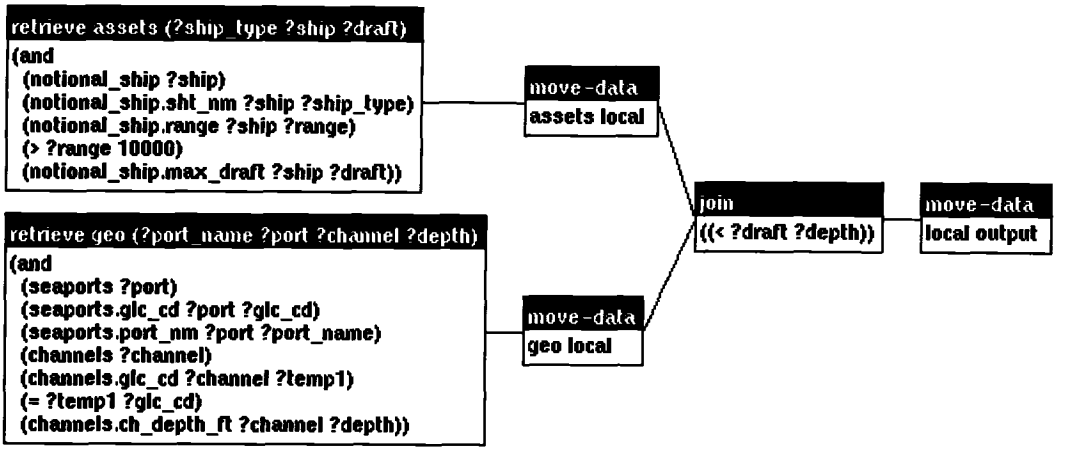

Figure 5: Parallel Query Access Plan

tractable. Ideally, our planning and learning systems would do this automatically.

References

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.

Anthony Barrett, Keith Golden, Scott Penberthy, and Daniel Weld. Ucpop user's manual (version 2.0). Technical Report 93-09-06, Department of Computer Science and Engineering, University of Washington, 1993.

Ken Currie and Austin Tate. O-plan: The open planning architecture. *Artificial Intelligence*, 52(1):49–86, 1991.

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

Craig A. Knoblock. Generating parallel execution plans with a partial-order planner. In *Artificial Intelligence Planning Systems: Proceedings of the Second International Conference (AIPS94)*, Chicago, IL, 1994.

Terry Landers and Ronni L. Rosenberg. An overview of multibase. In H.J. Schneider, editor, *Distributed Data Bases*. North-Holland, 1982.

R. MacGregor. The evolving technology of classification-based knowledge representation systems. In John Sowa, editor, *Principles of Semantic Networks: Explorations in the Representation of Knowledge*. Morgan Kaufmann, 1990.

M.P. Reddy, B.E. Prasad, and P.G. Reddy. Query processing in heterogeneous distributed database management systems. In Amar Gupta, editor, *Integration of Information Systems: Bridging Heterogeneous Databases*, pages 264–277. IEEE Press, NY, 1989.

Shashi Shekhar and Soumitra Dutta. Minimizing response times in real time planning and search. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, Detroit, MI, 1989.

David E. Wilkins. *Practical Planning: Extending the Classical AI Planning Paradigm*. Morgan Kaufmann, San Mateo, CA, 1988.

Qiang Yang, Dana S. Nau, and James Hendler. Merging separately generated plans with restricted interactions. *Computational Intelligence*, 8(4), 1992.

C.T. Yu and C.C. Chang. Distributed query processing. *ACM Computing Surveys*, 16(4):399–433, 1984.