A SAT Approach to Query Optimization in Mediator Systems [*]

Steven Prestwich
Cork Constraint Computation Centre
University College, Cork, Ireland
`s.prestwich@cs.ucc.ie`

Stéphane Bressan
School of Computing
National University of Singapore
`steph@comp.nus.edu.sg`

Abstract

Mediator systems integrate distributed, heterogeneous and autonomous data sources, but their effective use requires the solution of hard query optimization problems. This is usually done in one of two ways: when the set of data sources is fixed, their ordering into a feasible and efficient query is treated as a *join order* problem; when the set is not known, their selection from a superset is treated as a *set covering* problem. This two-phase approach is unlikely to find optimum queries, and we describe a generalised approach in which a feasible query is constructed from a subset of the available data sources. Under simple cost assumptions this can be encoded and solved as a SAT problem. Results on artificial benchmarks indicate that this is an interesting problem, with a combination of features that make it hard for both complete and incomplete search. We investigate both types of algorithm and a hybrid approach, with and without symmetry breaking constraints. The best combination turns out to be hybrid search without symmetry breaking.

1 Introduction

To effectively and efficiently leverage the growing amount of strategic information readily available online, modern decision support systems require direct access to this wealth of facts and figures. Yet the World Wide Web is merely a distributed collection of heterogeneous and autonomous data sources. It does not have the explicit schema expressed in a structured data model that top-down designed and centrally managed information would have.

In his seminal paper [3] Gio Wiederhold proposed the mediation architecture as a blueprint for the design of systems aimed at the intelligent integration of distributed, heterogeneous and autonomous sources. Most existing or proposed mediation systems (for example HERMES [14], TSIMMIS [2], Information Manifold [7], DISCO [15] and Context Interchange [1]) comprise the three components of the mediation architecture. *Facilitators* provide user and application programming interfaces that

[*]This research was partially funded by grant R-252-000-064-107 from the National University of Singapore.

enable structured querying of the data on the Internet and the World Wide Web. Each source may be a Web page, a Web service, or the front-end to an online database or application. The data provided by the sources may come in a variety of formats and may be accessible with various modalities. *Wrappers* provide the physical connectivity to the source as well as the first level of logical connectivity: they offer a view of the source in terms of the data model (in this paper we use a simple relational model) and query language (we use a domain calculus based query language) of the mediation system. Wrappers support the translation of queries written in the mediation system's structured language to the modes offered by the source, as well as the translation of information extracted from the source into the data model of the mediation system. The central components of this architecture are *mediators* that allow the optimization and processing of queries to the component sources via their wrappers.

In addition to the traditional tasks that a database query optimizer performs, three further issues must be handled by the optimizer of a mediator. Firstly, the sources may not be databases offering an unconstrained set of query capabilities, because wrappers provide limited-capabilities interfaces. For example a wrapper interfacing an online bookstore may only support queries that indicate the name of the author or the title of the book searched, because this is the only interaction authorized by the form-based Web interface. Therefore the optimizer must compose a plan under feasibility constraints [5, 9, 12, 14, 16]. Secondly, the optimizer must perform its optimization under cost models that estimate communication and source processing costs in a volatile environment, where none of these figures are precisely known nor can be controlled. For example (as we have all experienced as Web users) the response time of a query to various online bookstores may vary greatly from one Web site to another. It depends on various parameters that are difficult to predict, such as network traffic, the caching strategy of proxy servers, and the server-side's computing power. Little work has been done on developing effective cost models (see [16] for instance). Thirdly, the same piece of information may be available from various sources. For instance the ISBN number of a book may be obtained from several (but not all) online bookstores, or from the Web site agencies managing the International Standard Book Number System data. Therefore the optimizer must decide which combination of sources is most efficient.

The last issue is rarely addressed in conjunction with the cost-based optimization performed by optimizers in existing or proposed mediation systems. Rather it is left to the user or programmer, or treated separately in a pre-optimization phase based on simplistic heuristics, leaving the optimizer with a simpler planning problem. Obviously these strategies are sub-optimal. The challenge addressed in this paper is that of devising a model and techniques for the cost-based optimization of queries to mediator systems under limiting capabilities and with redundant sources. Our approach is to integrate the selection of data sources from a redundant set with their organisation into a feasible plan. In principle this should give optimum solutions, but the complexity of the problem may require modern modeling and search techniques. We present a formulation of the problem using a SAT approach, and preliminary results on artificial benchmarks, under a simple model in which the cost of a query plan is its length. An advantage of SAT modeling is that several highly optimized algorithms are immediately available for application, based on backtracking, local search and other approaches. This enables a rapid comparison between different types of algorithm. Another motive for applying the SAT approach is that it is sometimes very effective; for example SAT algorithms sometimes out-perform specialised planning algorithms [6].

2 Combining set covering with feasibility

The *databank problem* or selection of a file in a data bank is defined in [4] as follows. When some information has been requested, it is required to find which files in a data bank should be consulted to obtain it as cheaply as possible. Let S be a set of units of information asked for (set of tasks). Let E be the set of files which make up the data bank (set of means). A file $E_j \in E$ is then characterized by the set of units of information it contains, and by the cost c_j of using it (the time taken to consult it). The problem is then to find a minimum-cost *cover* of S. The set cover problem is NP-complete, even when all costs are unity (the *unicost* set cover problem). Another optimization problem that occurs in database and mediator applications is the *join order problem*. Given a *fixed* set of data sources, we must decide in which order they are to be consulted. The aim is to obtain a query plan that is *feasible* (sources must be ordered so that each is provided with its required inputs) and *efficient* (costs are to be minimised). This is NP-hard in general but can be solved in linear time when costs are ignored [16].

If neither the set of sources nor their best order is known, a two-phase approach may be used: solve the databank problem to obtain a set of sources, then solve a join order problem to assemble them into a feasible and efficient plan. Our approach subsumes unicost set covering (as shown in Section 4) but it does not subsume join order problems because its cost model is currently too simple. However, it does combine set covering with the notion of a feasible query plan. A data source is modeled by a subgoal H with *attributes* divided into inputs $i_1, i_2, \ldots$ and outputs $o_1, o_2, \ldots$, written $H(i_1, i_2, \ldots; o_1, o_2, \ldots)$. The inputs of a subgoal correspond to the data that must be provided to the source in order to obtain the data represented by the output. Note that a single source may correspond to several subgoals, for instance an online bookstore may correspond to both

- Bookstore(isbn; author, title) if a request can be posted through a form to find the title and authors of books given an ISBN number; and
- Bookstore(title; author, isbn) if a request can be posted through a form to list the authors and isbn numbers of books with a given title.

An ordered list of subgoals is referred to as a *plan*. A plan that satisfies [fails to satisfy] a condition **F** is *feasible* [*infeasible*]. The condition **F** is that for any input i_j in any subgoal $H(i_1, i_2, \ldots; o_1, o_2, \ldots)$ in the plan there exists at least one subgoal earlier in the plan with output i_j. The problem is to generate a feasible plan of minimum cost that provides one or more specified attributes as output. In this paper the cost of a query plan is its length. Besides being convenient for SAT encoding, this simple cost model can be applied to applications in which no cost information is available, for example in an internet application where the data sources are cookies.

To take a concrete example, suppose we have attributes $V = \{a, b, c, d, e, f, g, h\}$ and available subgoals

$$E = \{\ W(;a,b,c),\ X(b;d),\ T(b;g),\ U(g;h),\ V(h;d),\ Y(c;e),\ Z(d,e;f)\ \}$$

Note that subgoal W has no inputs: the plan always starts with at least one such subgoal. Note also that W's output a is not an input for any subgoal: such attributes may occur but are irrelevant. The problem is to find a plan that provides attribute f as output, that is with its only provider Z

occurring somewhere in the plan (typically in the final position). Two feasible plans for this example are:

$$P_1 = [\ W(;a,b,c),\ T(b;g),\ U(g;h),\ V(h;d),\ Y(c;e),\ Z(d,e;f)\ ]$$
$$P_2 = [\ W(;a,b,c),\ X(b;d),\ Y(c;e),\ Z(d,e;f)\ ]$$

It can be verified by inspection that condition **F** is satisfied: for example the input c in subgoal Y occurs earlier in the plan as an output of subgoal W. Both plans are minimal in the sense that deleting any subgoal from either gives an infeasible plan. For example deleting $X(b;d)$ from P_2 gives

$$[\ W(;a,b,c),\ Y(c;e),\ Z(d,e;f)\ ]$$

which is infeasible because attribute d violates condition **F**. Plan P_2 is optimum because no shorter plan exists.

3 Problem modeling

Given n available subgoals numbered $1 \ldots n$ (where $n \geq k$), the problem is to construct a feasible plan from a minimum subset. This can be decomposed into a constraint satisfaction problem for each plan length k. A major design decision is what to model as variables, values and constraints. We could model the dependencies between subgoals, but to force the subgoals to form a partial order we would require $O(n^3)$ transitivity clauses. Instead we model a plan as an ordered list of k subgoals.

Define $k \times n$ Boolean variables $v_{i,j}$ where $1 \leq i \leq k$ and $1 \leq j \leq n$. The meaning of $v_{i,j} = T$ is that the i^{th} subgoal of the plan is subgoal j. The constraints are as follows. No plan position can take more than one subgoal:

$$\neg v_{i,j} \vee \neg v_{i,j'} \quad (1 \leq i \leq k,\ 1 \leq j < j' \leq n)$$

Repeated subgoals are not allowed:

$$\neg v_{i,j} \vee \neg v_{i',j} \quad (1 \leq i < i' \leq k,\ 1 \leq j \leq n)$$

Assuming the existence of one or more subgoals q_1, $q_2 \ldots$ that each provide all the required attributes as outputs, we place one of these subgoals in the last position k of the plan:

$$v_{q_1,k} \ \vee \ v_{q_2,k} \ \vee \ \ldots$$

If no such subgoals exist then we can define one before SAT-encoding: $X(x_1, x_2, \ldots; d)$ where $x_1, x_2, \ldots$ are the required attributes and the dummy attribute d is the only attribute required from the plan. For the feasibility condition **F**:

$$\left.\begin{array}{rllll} \neg v_{i,p} \vee & v_{1,a_1} \vee & \ldots & \vee\, v_{i-1,a_1} \vee & \\ & v_{1,a_2} \vee & \ldots & \vee\, v_{i-1,a_2} \vee & \ldots \end{array}\right\} \quad (1 \leq p \leq n\ , 1 \leq i \leq k)$$

where $\{a_1, a_2, \ldots\}$ is the set of subgoals with output a, and a ranges over the inputs of subgoal p.

The model contains the following symmetry: for two adjacent subgoals in the plan, if the first one provides no input to the second one then their positions can be interchanged. We call these *independent* subgoals. In the example $X(b; d)$ and $Y(c; e)$ are interchangeable in P_2, which therefore has two equivalent representations. Similarly there are four representations of P_1:

$$\begin{array}{l}[\ W(;a,b,c),\ T(b;g),\ U(g;h),\ V(h;d),\ Y(c;e),\ Z(d,e;f)\ ] \\ [\ W(;a,b,c),\ T(b;g),\ U(g;h),\ Y(c;e),\ V(h;d),\ Z(d,e;f)\ ] \\ [\ W(;a,b,c),\ T(b;g),\ Y(c;e),\ U(g;h),\ V(h;d),\ Z(d,e;f)\ ] \\ [\ W(;a,b,c),\ Y(c;e),\ T(b;g),\ U(g;h),\ V(h;d),\ Z(d,e;f)\ ]\end{array}$$

One way to remove this symmetry is to add constraints forcing adjacent independent subgoals to take a lexicographical ordering, so that only the first version is allowed. (Note that this does not remove all ordering symmetries: for example two subgoals may not be independent under the above definition because one provides an input to the other, but they can still be exchanged because this input is also provided by an even earlier subgoal.) The symmetry breaking constraints are as follows:

$$\neg v_{i,j} \vee \neg v_{i+1,j'}$$

where $1 \leq i \leq k-1$ and j,j' are independent subgoals (the inputs of j and outputs of j' are disjoint, and vice-versa).

Symmetry breaking is a way of avoiding the generation of many representations of each solution, and can greatly reduce the size of a search space. It often greatly speeds up backtrack search, but in a recent paper [11] it was shown to have a contrary effect on local search. To further explore this result, in Section 4 we evaluate three search algorithms on our model both with and without symmetry breaking.

4 Experimental results

We shall evaluate three types of SAT solver. As a representative of complete (backtrack) search we use Chaff [8] (Z-Chaff version Z2001.2.17), possibly the fastest current SAT backtracker. As a representative of incomplete (local) search we use Walksat with the default (SKC) heuristic [13], which performs very well on many SAT benchmarks (and better than the Rnovelty heuristic on our problems). We also use an incomplete algorithm called Saturn, which is a recent implementation for 0-1 integer programs (including SAT) of the CLS algorithm [10]. CLS is a hybrid of local search and constraint propagation designed for large, highly structured problems.

We compare the three algorithms on artificial benchmarks. These combine the features of set covering and feasibility, and are specified by four integer parameters (A, S, C, L) where L is the plan length. For set covering we model S subsets of 3 distinct randomly chosen elements from the base set of A elements. There is a single subgoal:

$$\{\ G_{C+S+1}(a_{C+1}, \ldots, a_{C+A};)\ \}$$

which occupies the last plan position and creates a demand for prior subgoals in the plan to supply its inputs. The set of S available subgoals corresponding to the S subsets is:

$$\{\ G_i(x; a, b, c) \mid C+1 \leq i \leq C+S\ \}$$

A=C	S	L	vars	clauses	Chaff		Walksat		Saturn	
with symmetry breaking										
5	10	3	33	294	18	(0.0)	18	(0.0)	3	(0.0)
10	20	11	341	9726	202	(0.25)	12404	(0.5)	15899	(3.6)
15	30	17	782	34024	3278	(11)	3694947	(302)	371893	(19.4)
20	40	19	1159	67485	10059	(111)	20118045	(2579)	175504	(14)
25	50	30	2280	167642	aborted	(>1625)	—		—	
without symmetry breaking										
5	10	3	33	184	16	(0.0)	14	(0.0)	1	(0.0)
10	20	11	341	5556	388	(0.25)	3148	(0.1)	142	(0.06)
15	30	17	782	18616	45761	(718)	192646	(10)	324	(0.21)
20	40	19	1159	36291	aborted	(>1462)	323840	(25)	381	(0.34)
25	50	30	2280	88501	aborted	(>655)	—		95189	(78)

Figure 1: Results

where $C + 1 \leq a, b, c \leq C + A$ and $1 \leq x \leq C$, all distinct and chosen randomly. There are also C subgoals:

$$\{\ G_1(;a_1),\ G_2(a_1;a_2),\ G_3(a_2;a_3),\ \ldots,\ G_C(a_{C-1};a_C)\ \}$$

which supply the inputs x for the set covering-based subgoals, and also create a chain of feasibility requirements. Here is an example with $S = 3$, $A = 3$ and $C = 4$:

$$\begin{array}{ll}\{ & G_1(;a_1),\ G_2(a_1;a_2),\ G_3(a_2;a_3),\ G_4(a_3;a_4), \\ & G_5(a_2;a_{13},a_{10},a_{14}),\ G_6(a_4;a_{11},a_{14},a_{13}),\ G_7(a_1;a_{12},a_{11},a_{10}), \\ & G_8(a_{10},a_{11},a_{12},a_{13},a_{14};)\ \}\end{array}$$

The shortest feasible plan is $[G_1, G_2, G_5, G_7, G_8]$: G_8 takes its inputs from G_5 and G_7, which in turn need inputs a_1 and a_2. a_2 is provided by G_2, which requires a_1, which is provided by G_1. G_7 also needs a_1 which is already provided.

For these experiments we set $S = 2A = 2C$ and the plan length L to the smallest value for which a solution was found, in order to obtain hard instances. On all problems Saturn's noise parameter was set to 1 and Walksat's to 50%: results are not greatly improved by parameter tuning. All figures are means over 10 runs, except those for Chaff which is deterministic and can only be executed once per problem. Results are shown in Figure 1, where "—" denotes failure to find a solution after several hours, and "aborted" denotes that Chaff aborted the run, with the time after which it aborted given in brackets. The figures shown are decisions for Chaff, backtracks for Saturn, and flips for Walksat, with mean execution times in seconds (on a 300 MHz DEC Alphaserver 1000A 5/300 under Unix) shown in brackets.

On the problems with symmetry breaking Saturn scales best to large instances. Perhaps surprisingly, Walksat does not scale as well as Chaff. However, on pure set covering problems Walksat scales similarly to Saturn [results omitted for space reasons]: it appears to have difficulty with the feasibility condition. On the problems without symmetry breaking Saturn again scales best, but this time Walksat is second and Chaff scales very poorly. This shows the (unsurprising) benefits of symmetry breaking for backtrack search, but it also confirms a surprising result in [11]: that symmetry breaking can be very counter-productive for incomplete search. The results also show that on this class of problem hybrid search pays off. In summary, the best algorithm/model combination is the hybrid without symmetry breaking.

5 Conclusion

This paper described a new formulation of mediator system query optimisation, which is usually solved in two phases. By combining two hard problems into a single problem, the task of finding optimal or near-optimal queries should be greatly facilitated. The new problem was SAT-encoded and artificial benchmarks were designed. A simple cost function was used to enable SAT encoding, but we believe that the results will be useful as a guide to modeling and solving problems with more complex cost functions. Moreover, the cost function is not too trivial to have practical applications. In experiments:

- Backtrack search worked well only up to a point (a plan length of about 20 data sources on our instances), after which scalability became a problem.
- Local search had difficulty in solving instances in which feasibility plays a significant role.
- Symmetry breaking helped a complete algorithm but was very counter-productive in combination with two incomplete algorithms.
- Best results were obtained using an incomplete hybrid algorithm with a highly symmetric model.

A side benefit of this work is that it generated interesting new SAT benchmarks that appear to be intrinsically quite easy, yet are hard for both backtrack and pure local search. In future work we intend to obtain benchmarks from real applications.

References

[1] S. Bressan, C. Hian Goh, N. Levina, S. Madnick, A. Shah, M. Siegel. Context Knowledge Representation and Reasoning in the Context Interchange System. *Applied Intelligence* vol. 13 no. 2, 2000, pp. 165–180.

[2] H. Garcia-Molina et al. The tsimmis approach to mediation: data models and languages. *Journal of Intelligent Information Systems* vol. 8, 1997, pp. 117–132.

[3] G. Wiederhold. Mediators in the Architecture of Future Information Systems. *IEEE Computer*, March 1992, pp. 38–49.

[4] M. Gondran, M. Minoux. *Graphs and Algorithms*, Wiley-Interscience Series in Discrete Maths, Wiley-Interscience Editor, 1984.

[5] L. Haas, D. Kossman, E. Wimmers, J. Yang. An optimizer for heterogeneous systems with non-standard data and search capabilities. *Special Issue on Query Processing with Non-Standard Data, IEEE Data Engineering Bulletin*, 1997, pp. 37–43.

[6] H. Kautz, B. Selman. Pushing the Envelope: Planning, Propositional Logic and Stochastic Search. *Proceedings of the Thirteenth National Conference on Artificial Intelligence* vol. 2, MIT Press, 1996, pp. 1194–1201.

[7] A. Levy, A. Rajaraman, J. Ordille. Querying heterogeneous information sources using source descriptions. *Proceedings of the 22nd Conference on Very Large Databases*, 1996, pp. 251–262.

[8] M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang, S. Malik. Chaff: Engineering an Efficient SAT Solver. *Proceedings of the 39th Design Automation Conference,* Las Vegas, June 2001.

[9] Y. Papakonstantinou, A. Gupta, L. Haas. Capabilities-based query rewriting in mediator systems. *Proceedings of the International Conference on Parallel and Distributed Information Systems*, 1996.

[10] S. D. Prestwich. A Hybrid Search Architecture Applied to Hard Random 3-SAT and Low-Autocorrelation Binary Sequences. *Proceedings of the Sixth International Conference on Principles and Practice of Constraint Programming, Lecture Notes in Computer Science* vol. 1894, Springer-Verlag 2000, pp. 337–352.

[11] S. D. Prestwich. First-Solution Search with Symmetry Breaking and Implied Constraints. *Proceedings of the CP'01 Workshop on Modelling and Problem Formulation.*

[12] A. Rajaraman, Y. Sagiv, J. D. Ullman. Answering queries using templates with binding patterns. *Proceedings of the 14th ACM Symposium on Principles of Database Systems*, 1995, pp. 105–112.

[13] B. Selman, H. Kautz, B. Cohen. Noise Strategies for Improving Local Search. *Proceedings of the Twelfth National Conference on Artificial Intelligence*, AAAI Press 1994, pp. 337–343.

[14] V. S. Subrahmanian et al. Hermes: A heterogeneous reasoning and mediator system. http://www.cs.umd.edu/projects/hermes/overview/paper, 1996.

[15] A. Tomasic, R. Amouroux, P. Bonnet, O. Kapitskaia, H. Naacke, L. Raschid. The distributed information search component (disco) and the world-wide web. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 1997.

[16] R. Yerneni, C. Li, J. Ullman, H. Garcia-Molina. Optimizing Large Join Queries in Mediation Systems. *Proceedings of the International Conference on Database Theory*, 1999.