Tradeoff in Rule Induction for Semantic Query Optimization

Chun-Nan Hsu
Department of Computer Science and Engineering
Arizona State University
PO Box 875406,Tempe, AZ 85287
mailto:chunnan@asu.edu
http://www.isi.edu/sims/chunnan/

Abstract

Semantic query optimization (SQO) is a promising approach to the optimization of increasingly complex query plans in global information systems. The idea of SQO is to use semantic rules about data to reformulate a query into an equivalent but less expensive one. Since it is difficult to encode required semantic rules, a complete SQO system also includes a rule induction system and a rule maintainer. To maximize the net utility of learning, a rule induction system needs to learn those rules that are *effective* in reducing the query execution cost while *robust* against data changes to minimize the rule maintenance cost. This paper focuses on this tradeoff between effectiveness and robustness in the rule induction for SQO. The solution is to explicitly estimate the degree of the robustness of rules. The system can use the estimated robustness to make decisions to guide rule construction, guide rule repair, and control the size of a rule set. This paper also briefly reviews how robustness can be efficiently estimated and reports the initial experimental results.

Learning for Semantic Query Optimization

Semantic query optimization (SQO) (King 1981) is a promising query optimization technique for intelligent information mediators (Wiederhold 1992; Arens *et al.* 1993; Levy, Srivastava, & Kirk 1995) that integrate heterogeneous information sources because it can complement conventional query optimization techniques to overcome the heterogeneity and considerably reduce query execution cost. The essential idea of semantic query optimization is to use semantic rules about data to reformulate a query into a more efficient but semantically equivalent query.

Example 1: Suppose we have a query that retrieves the ship classes and the maximal draft of the ships in those classes which satisfy the following conditions: the ships in the class are capable of carrying containers, and their draft is less than 50 feet. This query is specified as follows:

```
  query(?ship_class,?draft):-
1:   ship_class(?ship_class,_,?draft,_,?ctnr),
2:   ship(_,?ship_class,_,_,?status),
3:   ?ctnr = "Y",
4:   ?status = "active",
5:   ?draft < 50.
```

In addition to the query, suppose the system possesses a set of semantic rules. Among them two rules are applicable to this query:

```
R1: IF ship(_,?class,?status,_,_) &
       ship_class(?class,_,?draft,_,_) &
       ?draft < 50
    THEN ?status = "active"
R2: IF ship_class(?class,_,_,_,?ctnr) &
       ?ctnr = "Y"
    THEN ship(_,?class,_,_,_)
```

`R1` states that *If the maximum draft of a ship is less than 50, then its status is active.* `R2` states that *If a ship class has container capability, then there must exist some ships that belong to that ship class in the database.*

Given these rules, a semantic query optimizer can derive a set of equivalent queries and search for the one that is the optimal. In this example, the optimizer infers that literal 4 and 2 are redundant and eliminates these literals from the query. This yields the optimized query as below:

```
 query(?ship_class,?draft):-
   ship_class(?ship_class,_,?draft,_,?ctnr),
   ?ctnr = "Y",
   ?draft < 50.
```

This new query is equivalent to the original query, because literal 4 is redundant according to `R1`, and from `R2` literal 2 is also redundant. Therefore, the optimized query will still return the same answer. Since the system does not need to evaluate the redundant comparisons and relational joins, executing the optimized query will save much query execution cost. □

A set of high utility semantic rules is crucial to the performance of a semantic query optimizer. Since it is difficult to encode sufficient semantic rules, researchers have proposed several approaches to rule induction for semantic query optimization (Siegel 1988;

Yu & Sun 1989; Shekhar *et al.* 1993; Hsu & Knoblock 1994). A rule maintenance approach is also necessary because the learned rules may become inconsistent with data after updates to the database, and the number of rules may grow so large that they may slow down the optimization and reduce the savings. Therefore, a complete semantic query optimization system should include three components — a query optimizer, a semantic rule learner and rule maintainer.

As other optimization problems, semantic query optimization needs to balance a variety of tradeoffs under the resource constraints. For example, the query optimizer must trade off the time spent for the optimization and the quality of the resulting query. Previously, Shekhar et al. (Shekhar, Srivastava, & Dutta 1988) presented an approach to the tradeoff in the query optimizer. This paper focuses on the tradeoff in the rule induction.

Tradeoffs between Effectiveness and Robustness

The rule induction problem for SQO is to learn a set of high utility semantic rules that maximize the net performance of the optimizer, that is, the learner must derive high utility rules that are effective in producing high savings for a wide range of queries while incurring minimal cost to be used. The cost to use semantic rules includes the storage space for the rules, the computation time to match and apply the rules during the optimization, and the cost to maintain inconsistent rules in the presence of database changes. Inconsistent rules are not useful for SQO because using inconsistent rules the optimizer may reformulate a query into a new query not equivalent to the original query and cause the system to produce incorrect results.

In real-world applications, database usage can be modeled as a stochastic sequence of queries and data modification transactions over time. If a learning approach can always learn invariant semantic rules [1] that are consistent with all possible database states [2] regardless of how a database changes, then the cost of maintaining inconsistent rules can be eliminated. However, it is prohibitive to guarantee that all the learned rules are invariants, because it is impossible for the learner to verify whether a rule is invariant without complete knowledge about the database application domain. A more practical solution to deal with data changes is to learn *robust* semantic rules. A semantic rule is robust if it is *unlikely* to become inconsistent in new database states after data changes. An invariant semantic rule is extremely robust.

However, robust rules may not be effective in cost reduction. A set of semantic rules is extremely effective if, for any query, it allows the optimizer to reformulate the query into the optimal equivalent query. Intuitively, an optimal equivalent query is the one that returns the same answer and can be executed with the lowest possible cost. If we take database changes into account, an optimal equivalent query in one database state might not be equivalent to a given query in another database state, especially when the semantic rules used to infer the equivalent queries are not invariant. Therefore, a set of effective rules might not be robust. In fact, semantic rules that "overfit" the data can usually produce large savings in query optimization, but they are useful only in one or two database states and learning these rules may increase rule maintenance cost. Therefore, the learner must balance these two factors — effectiveness and robustness — to maximize the net utility of learning.

Using Robustness Values for Resource Control

I have developed a general solution (Hsu 1996) to deal with the tradeoff in the rule induction for SQO. The solution is to develop an approach to estimating the degree of robustness of semantic rules (Hsu & Knoblock 1996). A query optimization system can use the estimated results to make decisions in the following manners.

Rule Pruning The rule induction system can apply the robustness estimation to guide the pruning of partially constructed rules. The rule pruning may include pruning the antecedent literals of rules or pruning low robust rules from a rule set. The basic idea is to search for a subset of antecedent literals to remove until any further removal will yield an inconsistent rule. The approach applies a beam search strategy to retain the top robust rules for each set of pruned rules with the same length and then selects a subset of pruned rules with a good combination of length and robustness based on how often the database changes. As a result, for each set of equally short rules, the pruner will search for the rule that is as robust as possible while still being consistent. Pruning antecedent literals also increases the applicability and thus the effectiveness of a learned rule. Therefore, the resulting rules will be highly robust and widely applicable. This approach has been implemented into a system and incorporated into a query optimization system (Hsu & Knoblock 1996; Hsu 1996).

[1] Invariant semantic rules are also referred to as semantic integrity constraints.

[2] A database state at a given time `t` is the collection of the instances present in the database at the time `t`.

Rule repair We can also apply the robustness estimation approach to rule maintenance in a manner similar to the rule pruning. When an inconsistent rule is detected, the rule maintenance system may propose and search through a set of rule repair operators (e.g., modify a condition) to fix the rule. The maintenance system can use the estimated robustness of the resulting partially repaired rules to search for the best sequence of repair operators so that the repaired rule is more robust than the original one. Since the rules are increasingly robust, eventually the need of rule repair can be eliminated.

Controlling the size of the rule set A large rule set may increase the match cost and slow down the optimization. Hence, it is beneficial to control the size of the rule set. During the rule pruning, the system may adjust the beam size of the search to control the number of generated rules. The rule maintainer can remove the learned rules whose robustness is below a threshold of minimum robustness to control the size of the rule set. The maintainer may also use the statistics of rule utility coupled with the estimated robustness for this purpose.

Estimating Robustness

We have defined that a rule is robust against database changes if it is unlikely to become inconsistent after database changes. This can be expressed as the probability that a database is in a state consistent with a rule and estimated by the ratio between the number of all possible database states and the number of database states consistent with a rule. However, this is based on an unrealistic assumption that all database states are equally probable. Also, since the number of database states is intractably large even for a small database, this may not be a good estimate for robustness. Instead, we define the notion of the robustness against database changes by defining the robustness of a rule r in a given database state d as

$$\mathrm{R}obust(r|d) = \Pr(\neg t|d) = 1 - \Pr(t|d),$$

where t represents the data modification transactions on d that invalidate r. This definition localizes the database states of concern to those that are accessible from a given database state, and thus allows a rule discovery system to estimate the robustness efficiently. The robustness estimation problem otherwise would be intractable because the system must estimate combinatorial numbers of database states that are inconsistent with a rule.

The robustness estimation approach estimates probabilities of rule invalidating transactions in a relational database environment. This approach decomposes the probability of a transactions into local probabilities based on a Bayesian network model of database modifications as illustrated in Figure 1. The local probabilities are estimated using Laplace law or m-probability. Users do not need to provide additional information for the estimation because the estimator can utilize information such as transaction logs, database schema, and ranges of attribute values that are available from a database management system. Even if the information is incomplete or unavailable, the approach can still derive a reasonable estimation. The approach can accurately estimate the robustness of Horn-clause rules.

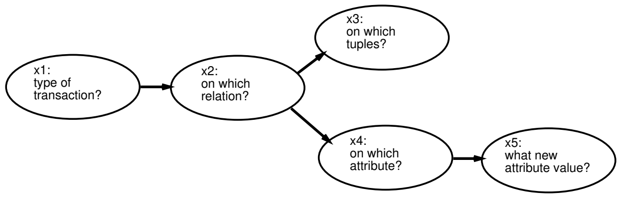

Figure 1: Bayesian network model of transactions

Experimental Results

This section describes the initial empirical evaluation on the proposed approaches to dealing with the tradeoff between effectiveness and robustness. For this purpose, the approach were implemented in a learning system and a query optimization system and incorporated with an information mediator that integrates heterogeneous databases.

The evaluation consists of three experiments. The first experiment is to evaluate the effectiveness of the rules generated by the learning approach when they are applied to optimize queries in a static database state. The experiment compares the optimization performance produced by the learned rules with handcoded rules.

The second experiment is to evaluate our approach to dealing with database changes. Since we do not have access to a sequence of data modification transactions that is sufficiently long to simulate real-world database usage, we cannot fully demonstrate the net savings yielded by applying our learning and optimization approach. However, by showing the accuracy of the robustness estimation, it suffices to affirm that a learner can minimize its effort in rule learning and maintenance while provide high utility rules for the optimizer.

The third experiment examines the interaction between effectiveness and robustness of a semantic rules. The experiment compares a variety of properties of the learned rules — optimization performance, robustness and applicability.

Environment for the Experiments The rule induction system BASIL[3] is an implementation of the learning approach to the acquisition of high utility semantic rules for SQO. BASIL learns semantic rules for PESTO,[4] an implementation of the query plan optimization approach. PESTO uses semantic rules learned by BASIL to optimize query plans for an information mediator. These systems are developed to empirically evaluate the approaches developed in this research (see (Hsu 1996) for more information on these systems). They are incorporated with the SIMS information mediator (Arens *et al.* 1993; Knoblock, Arens, & Hsu 1994; Arens, Knoblock, & Shen 1996). SIMS applies a variety of AI techniques to build an integrated intelligent mediator between users and distributed, heterogeneous multidatabases so that users can access those databases without knowing the implementation details such as their locations, query languages, platforms, etc. SIMS invokes PESTO to optimize query plans, and PESTO in turn invokes BASIL to learn the required semantic rules.

SIMS takes as input a query expressed in the LOOM knowledge representation language (MacGregor 1990), which is also used as the representation language to build an integrated model of databases. To optimize queries for SIMS, PESTO has a component to translate a LOOM subquery into an internal representation similar to Datalog to facilitate optimization, and a component to translate the result back to LOOM. The semantic rules are expressed in the same internal representation. By attaching different translation component, PESTO can optimize queries in other query languages. BASIL uses the same internal representation to express the semantic rules it learns. When it needs to read data or information about database (e.g., schema), BASIL sends a query to SIMS to obtain the required information.

For the purpose of our experiments, SIMS is connected with two remote ORACLE relational databases via the Internet. These databases originally are part of a real-world transportation logistic planning application. Table 1 summarizes the contents and the sizes of these databases. Together with the databases, there are 29 sample queries written by the users of the databases. We also have 3 queries written for the purpose to test different functionalities of the SIMS query planner, and 4 queries to test PESTO, especially to test its ability to detect null queries (i.e., queries that return an empty set). That is a total of 36 queries. Among these 36 queries, 18 are multidatabase queries that require access to multiple databases to retrieve the answer. To train the learner, BASIL, 23 queries are selected to serve as the training queries. The selection is based on the similarity of queries. Because we found that BASIL learns nearly identical sets of rules using similar queries, to save experimentation time, we remove some similar queries from the training set. In addition to the learned rules, PESTO uses 271 range facts compiled from the databases for the optimization. SIMS, PESTO and BASIL were running on a Sun SPARC-20 workstation during the experiments. We synthesized 123 sample transactions that represent possible transactions of the experimental databases based on the semantics of the application domain to evaluate the accuracy of the robustness estimation. The set of transactions contains 27 updates, 29 deletions and 67 insertions, a proportion that matches the likelihood of different types of transactions in this domain.

Effectiveness of Learned Rules The first experiment is to evaluate whether the learning approach can generate effective rules for cost reduction in a given database state. This experiment applies a *k-fold cross validation* (Cohen 1995) to test the effectiveness of the learned rules. The 23 training queries were randomly divided into four sets, three of them contain 6 queries, and one contains 5 queries. For each set of queries, BASIL took the remaining three sets of queries as training queries to learn a set of semantic rules. The selected set of queries was combined with the 13 additional queries to form the test set of queries. Next, SIMS took the test set as input and invoked PESTO to optimize the queries using the learned semantic rules. After collecting performance data, the learned rules were discarded and the process repeated for the next set of queries. The experiment thus generated four sets of performance data.

Prior to this experiment, we have hand-crafted a set of 112 semantic rules to demonstrate the effectiveness of the query plan optimizer PESTO, These rules were carefully designed after several iterations of debugging and modifications to allow the optimizer to explore as much optimization opportunity as possible for the sample queries. We report the optimization performance produced by the hand-coded rules for the purpose of comparison.

The collected performance data contains the total elapsed time of each query execution, which includes the time for database accesses, network latency, as well as the overhead for semantic query optimization. To reduce inaccuracy due to the random latency time in network transmission, all elapsed time data are obtained by executing each query 10 times and then computing their medians. Then for each query, the percentage time savings are obtained by computing the

[3]BAyesian Speedup Inductive Learning.

[4]Plan Enhancement by SemanTic Optimization.

Databases	Contents	Relations	Tuples	Size(MB)	Server
Geo	Geographical locations	15	56124	10.48	HP9000s
Assets	Air and sea assets	16	4881	0.51	Sun SPARC 4

Table 1: Sample databases in a transportation logistic planning domain

ratio of the total time saved due to the optimization over the total execution time without optimization.

Table 2 shows the average of the savings for all queries, the average of savings for multidatabase queries and the standard deviations. The data shows that the learned rules can produce a significant savings on the test queries, with a ten percent higher savings for multidatabase queries. The data also shows that the learned rules outperform hand-coded rules in all four tests. We note that some of our test queries are already very cost-effective, and there is not much room for optimization for those queries. But for some expensive multidatabase queries, the savings can reach as high as 70 to 90 percent.

Accuracy of Robustness Estimation This experiment evaluates the accuracy of robustness estimation so as to establish the claim that using the robustness estimation allows a learner to minimize the cost of dealing with database changes. The experiment design can be outlined as follows: train BASIL to learn a set of rules and estimate their robustness, use the 123 synthesized data modification transactions to generate a new database state, then check if high robust rules have a better chance to remain consistent with the data in the new database state.

To investigate the relation between the estimated robustness of rules and their consistency status in the new state, We classified the discovered rules into four robustness groups, according to their probabilities of consistency after the completion of 123 transactions. These probabilities were derived from their estimated robustness. The classification was made such that if the probability of consistency of a rule is greater than 0.75 then it is classified as a "high" robust rule, if the probability is between 0.75 to 0.50 then it is "medium" robust, if the probability is between 0.50 to 0.25 then it is "low" robust, otherwise, it is considered a "very low" robust rule.

In this experiment, BASIL was adjusted to exhaust its search space during the rule discovery and generated 355 rules. Meanwhile, BASIL estimated the robustness of these rules. We used all 23 training queries to train BASIL, and resulted in 355 rules. After generating the rules and collecting their robustness, we applied the set of 123 transactions to the two relational databases connected to SIMS and generate a new database state. Next, we checked the consistency of all 355 rules and identified 96 inconsistent rules in the new database state. Table 3 shows the number of rules in each levels of robustness against the number of actual consistency of rules. We performed a statistic significance test on the result in the table. Since we obtain $\chi^2 = 19.4356$ from this table, and under the null hypothesis that the consistency of a rule and its estimated robustness are independent, the probability to get a χ^2 value this high is less than 0.01, we conclude with a 99 percent confidence that the robustness estimation accurately reflects the likelihood of whether a rule may become inconsistent after data modification transactions.

In order to evaluate the predictive power of the robustness estimation, we define two measures

$$\text{recall} = \frac{|I \cap L|}{|I|}$$
$$\text{precision} = \frac{|I \cap L|}{|L|}$$

where I is the set of inconsistent rules and L is the set of rules that are estimated as likely to become inconsistent. The definitions are analogous to their definitions in natural language processing and information retrieval research. Intuitively, *recall* indicates the proportion of inconsistent rules being identified as likely to become inconsistent rules, and *precision* indicates the proportion of the estimatedly low robust rules that actually become inconsistent.

Consider that a threshold for low robust rules is set to the level "high," that is, if the estimated probability of consistency for a rule is less than 0.75, then it is predicted to become inconsistent after 123 transactions. From Table 4, this threshold produces a recall of 92.7 (= 89 / 96) percent and a precision of 28.89 (= 89 / 308) percent. That is, with this threshold, BASIL can accurately point out 92.7 percent of inconsistent rules. But on the other hand, among all those rules that are classified as likely to become inconsistent, only 28.89 percent actually become inconsistent. This is not surprising because the robustness estimation may overestimate the probability of invalidating transactions of a rule in situations where enumerating all possible invalidating transactions is too expensive. In fact, by raising the threshold, we can obtain a higher recall while maintain the precision to be around

	Test 1	2	3	4	Average savings	hand-coded rules
All	28.99%	31.60%	33.94%	29.86%	31.07%	25.84%
Multidb	39.43%	42.51%	42.61%	39.63%	41.05%	36.19%
# of Rules	101	119	106	118	111	112
opt time (s)	0.038	0.047	0.041	0.054	0.045	0.044

Table 2: Performance data of learned rules and hand-coded rules

	Consistent	Inconsistent	Total
high	40	7	47
medium	49	13	62
low	19	22	41
very low	151	54	205
Total	259	96	355

Table 3: The joint distribution of the actual and estimated robustness

28 percent. For example, if we set 0.95 as the threshold, then we can obtain a high recall of 98.95 percent, and a precision of 28.27 percent. Consequently, since the robustness estimation can accurately point out low robust rules, by properly adjusting the threshold, the estimated robustness values can provide the sufficient information for rule learning and maintenance to deal with database changes.

Effectiveness versus Robustness Intuitively, a low robust rule that expresses specific data regularity in a given database state might produce a high cost reduction, while a high robust rule such as an integrity constraint on the gender of pregnant patients in a hospital information system might not be effective for query optimization. However, there is no empirical data that verifies this intuition. This section describes an empirical study of the interaction between effectiveness and robustness of semantic rules. For the purpose of this study, BASIL uses the four robustness levels as in the previous experiment to filter learned rules into four different levels of robustness. We designed three experiments to compare the utility of the learned rules. The first experiment compares their average savings. The second experiment compares the converging rate of the coverage of the learned rules. The third experiment verifies our assumption on the interaction between the robustness, length and applicability of the learned rules.

We note that it is possible that an individual rule yields high cost reduction together with a set of rules but low with another set. Since how much an individual rule can contribute to the cost reduction can only be determined in the context of the entire rule set used for the optimization, we do not have any experiment to compare the effectiveness and robustness on an individual rule basis.

In the first experiment, for each level of robustness threshold, BASIL was modified so that it would discard a partially pruned rule if its estimated robustness was below the threshold during the rule pruning search. Then we applied a k-fold cross-validation as described earlier to obtain the average savings produced by the learned rules. Table 5 shows the average performance data with the standard deviations. The data show that the rules learned with the robustness thresholds may not produce savings as high as those with no robustness threshold. However, since there are more than twice as many rules in the "very low" (i.e., no threshold) case as in other cases, the low savings might be ascribed to the lack of sufficient number of rules. It is remarkable that using the set of 28.5 very high robust rules can still produce a 12.54 percent savings.

We applied the incremental k-fold cross-validation to obtain the data on the converging rate for different robustness thresholds. Figure 2 shows the plot of the data. Interestingly, the four curves for different thresholds have almost the same shape, that is, they converge at about the same rate. But with higher thresholds, the number of optimized queries is smaller than the case where no robustness threshold is applied.

Our rule pruning approach use length to measure the applicability of a rule and we assume that robustness may also interact with applicability. The next experiment attempts to verify this hypothesis. In this experiment, PESTO used the 355 rules learned for the previous experiment to optimize all of our 36 queries. We

	Consistent ($\neg I$)	Inconsistent (I)	Total	
High robust ($\neg L$)	40	7	47	
Low robust (L)	219	89	308	precision = 28.89%
Total	259	96	355	
	recall =	92.70%		

Table 4: The joint distribution of the actual and estimated robustness

	Average savings		Average # of rules		Avg opt. time (s)	
high	12.54%	s=1.89%	28.5	s=6.45	0.0287	s=0.008
medium	13.11%	s=2.71%	50.25	s=10.24	0.0393	s=0.010
low	13.54%	s=0.93%	63.0	s=11.1	0.0362	s=0.004
very low	31.07%	s=2.20%	111.0	s=8.91	0.038	s=0.038

Table 5: Performance data of rules with different robustness thresholds

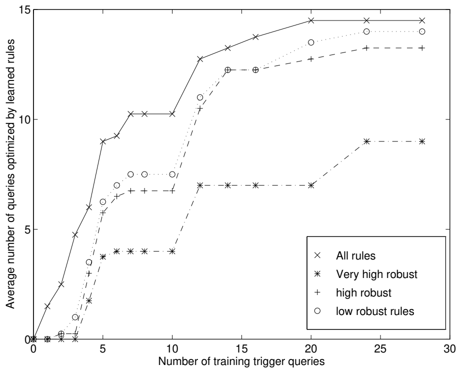

Figure 2: Rule coverage rates for rules of different robustness levels

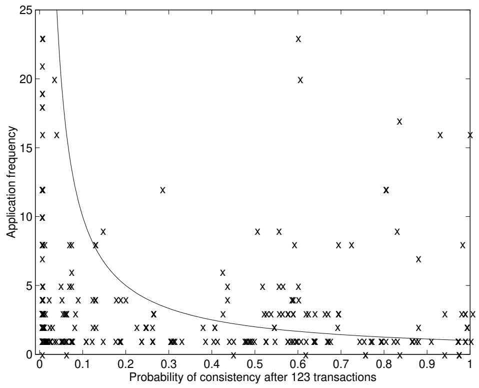

Figure 3: Relation between application frequency and estimated robustness

counted the application frequency for each rule, that is, how many times a rule is located as an applicable rule during the query optimization. Based on the data, we produced a scatterplot to visualize the relation between the the estimated probability of consistency and the applicability of a rule, as shown in Figure 3. As we expected, for most of rules, the probability of consistency is inversely proportional to their application frequency, because a high density of the rule population is distributed below the curve $y = 1/x$. We note that there is a significant population of rules positioned on the right-upper corner — they are both effective and robust. Figure 4 shows the scatterplot of the length of rules against their application frequency. The plot suggests that with few exceptions, widely applicable rules are short, but short rules are not necessarily widely applicable.

Conclusions

This paper has discussed a tradeoff between effectiveness and robustness in the learning problem of semantic query optimization and briefly described a solution to balance the tradeoff. The solution is to explicitly estimate the degree of robustness of the learned semantic rules. The estimation approach can be applied in rule pruning during the learning stage, in rule repair, and in evaluating the performance of the learned rules to control the size of the rule set. As a result, the system can effectively use the storage space and reduce rule maintenance cost. To more accurately eval-

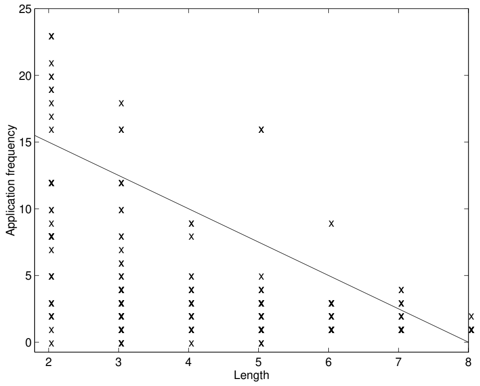

Figure 4: Relation between application frequency and length of rules

uate this approach, we plan to conduct a large scale long term experiment on real-world information systems and compare the net savings of applying the robustness estimation with other approaches.

Acknowledgements

The research reported here was supported in part by the National Science Foundation under Grant No. IRI-9313993 and in part by Rome Laboratory of the Air Force Systems Command and the Advanced Research Projects Agency under Contract No. F30602-94-C-0210 This work was partly done while the first author worked as a graduate research assistant at USC/Information Sciences Institute.

References

Arens, Y.; Chee, C. Y.; Hsu, C.-N.; and Knoblock, C. A. 1993. Retrieving and integrating data from multiple information sources. *International Journal on Intelligent and Cooperative Information Systems* 2(2):127–159.

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.*

Cohen, P. R. 1995. *Empirical methods for artificial intelligence.* Cambridge, MA: The MIT Press.

Hsu, C.-N., and Knoblock, C. A. 1994. Rule induction for semantic query optimization. In *Machine Learning, Proceedings of the 11th International Conference(ML-94).* San Mateo, CA: Morgan Kaufmann.

Hsu, C.-N., and Knoblock, C. A. 1996. Discovering robust knowledge from dynamic closed-world data. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96).* Portland, Oregon: AAAI Press.

Hsu, C.-N. 1996. *Learning Effective and Robust Knowledge for Semantic Query Optimization.* Ph.D. Dissertation, Department of Computer Science, University of Southern California.

King, J. J. 1981. *Query Optimization by Semantic Reasoning.* Ph.D. Dissertation, Stanford University, Department of Computer Science.

Knoblock, C. A.; Arens, Y.; and Hsu, C.-N. 1994. Cooperating agents for information retrieval. In *Proceedings of the Second International Conference on Intelligent and Cooperative Information Systems.*

Levy, A. Y.; Srivastava, D.; and Kirk, T. 1995. Data model and query evaluation in global information systems. *Journal of Intelligent Information Systems, Special Issue on Networked Information Discovery and Retrieval* 5(2).

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

Shekhar, S.; Hamidzadeh, B.; Kohli, A.; and Coyle, M. 1993. Learning transformation rules for semantic query optimization: A data-driven approach. *IEEE Transactions on Knowledge and Data Engineering* 5(6):950–964.

Shekhar, S.; Srivastava, J.; and Dutta, S. 1988. A formal model of trade-off between optimization and execution costs in semantic query optimization. In *Proceedings of the 14th VLDB Conference.*

Siegel, M. D. 1988. Automatic rule derivation for semantic query optimization. In Kerschberg, L., ed., *Proceedings of the Second International Conference on Expert Database Systems.* Fairfax, VA: George Mason Foundation. 371–385.

Wiederhold, G. 1992. Mediators in the architecture of future information systems. *IEEE Computer.*

Yu, C. T., and Sun, W. 1989. Automatic knowledge acquisition and maintenance for semantic query optimization. *IEEE Trans. Knowledge and Data Engineering* I(3):362–375.