Optimal CQ Reformulation is NP-Hard

Abhijeet Mohapatra and Michael Genesereth

Stanford University, Stanford, CA - 94305, USA
{abhijeet, genesereth}@cs.stanford.edu

In this article, we analyze the time complexity of computing an optimal reformulation of conjunctive queries subject to a linear growth in storage space. We can limit the search space of potential reformulations of a conjunctive query by applying the *Subgoal Theorem* and the *Projection Lemma*.

Theorem 1 (Subgoal Theorem). *For any reformulation q' of a conjunctive query q, there is an equally good or better reformulation q'' of q in which the bodies of all views are subsets of the body of the original query.*

The value of the Subgoal Theorem is that it presents an easy method for us to enumerate the view definitions used in useful reformulations, i.e., by considering subsets of the subgoals in the original query and no more. We can further narrow the set of potential reformulations by taking advantage of the linear storage growth constraint.

Definition 1. *A projection view is a view in which the body of a view definition includes at least one subgoal that contains all of the variables in the head.*

Lemma 1 (Projection Lemma). *In any reformulation that satisfies the linear growth constraint, every view must be a projection view.*

As a consequence of the Subgoal Theorem and the Projection Lemma, we need only consider projection views that are defined in terms of subgoals. Unfortunately, despite these improvements, we show that the optimization problem is still NP-Hard in the number of subgoals and free variables in the input query. Our proof strategy is as follows. First, we prune the space of potential reformulations by only considering reformulations that preserve the *connectivity* among the head variables. We call such reformulations as *dpp-transforms*. Then, we show that computing an optimal *dpp-reformulation* is NP-Hard by proving that this problem is isomorphic to the Steiner Tree problem [1].

Definition 2. *A p-transform of a query is a rewriting of a the query using projection views, where the body of the views are identical to the body of the query.*

Given a query, two variables groups $\bar{X}$ and $\bar{Y}$ are *connected* with respect to the query if any of the following conditions hold.

- $\bar{X} = \bar{Y}$. (path length$_{\bar{X}\bar{Y}} = 0$)
- The body of the query contains an atom with $\bar{X}$ and $\bar{Y}$, i.e., the atom is of the form $r_i(\bar{X}, \bar{Y}, \bar{Z})$. (path length$_{\bar{X}\bar{Y}} = 1$)
- There exists variable groups $\bar{U}$ and $\bar{V}$ such that the pairs: $\bar{X}$ and $\bar{U}$, $\bar{U}$ and $\bar{V}$, and $\bar{Y}$ and $\bar{V}$ are connected. (path length$_{\bar{X}\bar{Y}} = 2 +$ path length$_{\bar{U}\bar{V}}$)

Definition 3 (dpp-transform). *A p-transform q' of a query q is a dpp-transform iff for every pair of head variables X and Y in q such that X and Y are connected w.r.t q, the variables X and Y are also connected w.r.t q' by a unique path.*

Lemma 2. *Every dpp-transform of a query is a reformulation.*

Proof. We prove the above lemma using induction on the size of the path between head variables. The scenario where a supplied query consists of only one head variable case is uninteresting because the materializing the query itself does not violate the linear storage constraint, and is the optimal reformulation. Therefore, we focus on the case where a supplied query has two head variables.

Base case. In this case, there exists a shortest path of length one between the head variables, say, X and Y, i.e., there exists an atom of the form $r(X, Y)$. We can materialize the query without violating storage constraint. If there exists a shortest path of length two, e.g. $q(X, Y)$:- $v_1(X, A)$, $v_2(A, Y)$, $v_3(X, B)$ $v_4(B, Y)$, we may either drop A or B to get an equivalent reformulation.

Inductive hypothesis. If the shortest path between the head variables is of length $\leq n$, the lemma holds, and we may eliminate variables preserving a unique path, and have an equivalent reformulation.

Inductive step. Suppose the shortest path between the head variables, say, X and Y is of length $n + 1$. Let N_y be the set of nodes $= \{A_1, A_2, \ldots, A_k\}$ that are directly connected to Y in such paths. Consider a rewriting of the dpp-transform where every view of the form $v(Y, \bar{A}, \bar{Z})$ where $\bar{A}(\subseteq N_y)$, is replaced by a view $t_v(Y, A_1, A_2, \ldots, A_k, \bar{Z})$. Clearly, this rewriting is equivalent to the dpp-transform. The shortest path between X and Y in the rewriting is n. Therefore, we can apply our induction hypothesis to prove the lemma.

We can extend our proof to queries with more than two variables by applying induction on the number of head variables. □

In the following, we focus on computing optimal dpp-transforms of queries with binary subgoals, and show that the time taken is NP-Hard in number of subgoals and free variables in the body.

Theorem 2. *For conjunctive queries with binary subgoals, the time taken to compute an optimal dpp-transform is NP-Hard in the number of subgoals and free variables in the query.*

Proof. We prove NP-Hardness by showing that the problem of computing an optimal dpp-transform is *isomorphic* to the Steiner Tree problem, which is NP-Complete [1]. Given an undirected graph $G(V, E)$, a subset of the vertices R called terminal nodes, the Steiner Tree is there a subtree $G'(V', E')$ of G such that $R \subseteq V'$, and there does not exist a subtree $G''(V'', E'')$, where $R \subseteq V''$ and $\mid E'' \mid < \mid E' \mid$.

Given an instance of a Steiner Tree problem, we *reduce* it to an instance of finding an optimal dpp-transform as follows.

- We label the terminal nodes as $X_1, X_2, \ldots, X_{|R|}$. We label the edges where at least one of the end-points is a non-terminal node as $A_1, A_2, \ldots, A_{|E|}$.
- To every pair of edges A_i, A_j incident on a non-terminal node, we assign a relation, say $r_m(A_i, A_j)$.
- To every edge incident A_i on a terminal node X_j, we assign a relation, say $r_k(A_i, X_j)$.
- To every between two terminal nodes X_i, X_j, we assign a relation, say $r_l(X_i, X_j)$.
- Let ϕ be the conjunction of the above relations. The output dpp-transform instance is the following query.

$$q(X_1, X_2, \ldots, X_{|R|}) \text{ :- } \phi$$

Let q' be an optimal dpp-transform of q. We can leverage the definition of q' to construct the Steiner Tree of G as follows. We construct G' by removing the edges in G that correspond to the variables that are present in the definition of q, but missing in q'. Since q' is a dpp-transform, there exists a unique path between any two head variables X_i and X_j. As a result, there is a unique path in G' between any two nodes. Therefore, G' is a tree. G' must be a Steiner Tree; if not, then we could obtain a better dpp-transform by starting with the Steiner Tree and applying the reduction outlined above.

Given a conjunctive query q with binary subgoals, we *reduce* it to an instance of Steiner Tree problem as follows.

- We create a node for every variable in the query.
- For every subgoal, say, $r(X, Y)$, we create an edge between nodes in the graph corresponding to X and Y.
- The set of terminal nodes R is the set of nodes corresponding to the head variables in the query.

Consider a Steiner Tree of $G'(V', E')$ of G. We can use G' to construct an optimal dpp-transform q' of q as follows. We replace each subgoal $s_i(X, Y)$ of q using a view $v_i(X, Y)$ where v_i is defined as $v_i(X, Y)$:- body of q. We remove all views of the form $v_k(X_i, X_j)$ if there is no edge between the nodes corresponds X_i and X_j in G'. We drop all variables X_i such that the node corresponding to X_i does not appear in G'. Since G' is a tree, there is a unique path between any two head variables in q'. Hence, q' is a dpp-transform. Suppose, there is a different dpp-transform q'' with fewer free variables than in q', then we can construct a small

subtree G'' that spans the terminal nodes R by applying the above reduction to q''.

Since the Steiner Tree problem is NP-Complete [1], the problem of computing an optimal dpp-transform is NP-Hard as well. □

References

1. Garey, M.R., Johnson, D.S.: Computers and Intractability; A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York, NY, USA (1990)