Efficient Recursive Query Processing using Wavefront Methods

by

C.T. Yu and Weining Zhang

Department of Electrical Engineering & Computer Science
University of Illinois at Chicago
Chicago, Illinois 60680

Abstract

In this paper, we study the optimization of linear recursive queries using wavefront methods. The following results are obtained: (i) In spite of seemingly reasonable approach of the wavefront methods, certain linear recursive queries are not processed efficiently or correctly. (ii) A characterization of the expressions generated by linear recursive rules is given. Properties of the expressions will be useful for efficient processing of linear recursive queries. (iii) Conditions for efficient processing of linear recursive rules using wavefront methods with the properties given in (ii) are provided.

2. Wavefront Methods:

Considering "typical" linear recursive rules of the form:

\[ r_e : R(x_1, \ldots, x_n) := F(x_1, x_2, \ldots, x_m), \ m \geq n \]
\[ r_r : R(x_1, \ldots, x_n) := W(y_1, \ldots, y_p) \bigcap \bigcap_{i} R(x_1, \ldots, x_n) \bigcap \bigcup_{i} \phi(x_i) \]

where \( R(x_1, \ldots, x_n) \) is the recursive predicate in the head of the recursive rule \( r_r \) and is denoted by \( R_h \); \( R(x_1, \ldots, x_n) \) is the recursive predicate in the body of rule \( r_r \) and is denoted by \( R_b \); \( W, V, F \) are base relations stored in the database; \( r_e \) is the exit rule.

Wavefront methods [HaLu, HeNa] have been suggested for the rules given above. The essence of the wavefront methods is to express the partial solution at the \((i+1)\)th iteration, \( PS_{i+1} \), in terms of some expressions computed at the \(i\)th iteration, so that redundancy in computation is reduced. For example, in the Single Wavefront method [HaLu, HeNa],

\[ W^{i+1} F V^{i+1} = (W^i) W F V^{i+1} \]

During the computation of \( PS_i \), the result of \( W^i \) will be saved and used for the computation of \( PS^{i+1} \). In the central wavefront method [HaLu],

\[ W^{i+1} = W(W^i F V^{i+1}) V = W(PS^i) V \]
Thus, PS\textsuperscript{i+1} can be obtained by joining W and V to PS\textsuperscript{i}. Usually, one or more of the parameters in R are instantiated.

3. Assumptions:

The analysis in this paper is based on the following assumptions. (The definitions of non-distinguished and distinguished variables will be given in Section 4).

(1) Arguments are function free.
(2) If a subset of distinguished variables appear at the same set of positions in both R\textsubscript{h} and R\textsubscript{b} of r\textsubscript{r}, then the position taken by each variable in the subset must be the same in R\textsubscript{h} and R\textsubscript{b} i.e. no non-trivial permutation is allowed.
(3) There is no repeated occurrences of any variable in R\textsubscript{h} of r\textsubscript{r}.
(4) A variable appearing in the head of the recursive rule must appear in the body of the rule.
(5) In the recursive rule, all non-distinguished variables are in some base predicates.

Assumptions (1) - (3) are similar to those used in [Ioan]. Assumption 4 is known as range restricted [Reit, BaRa]. Assumption 5 is not really required. It can be eliminated with minor changes in the results stated in later sections.

4. Weaknesses of the Wavefront Methods and some Remedies:

(i) Avoid Unnecessary Cartesian Products

As an illustration that previous techniques do not handle general linear recursive queries properly, consider the following example.

Example 1: The rules are:

\[ r_{e} : R(x,y,z) \leftarrow F(x,y,z) \]
\[ r_{r} : R(x,y,z) \leftarrow W(x,s)R(t,y,s)V(t,z) \]

and the query R(a,?,?) (i.e. those (y,z) satisfying R(a,y,z) should be returned).

It is easy to verify that the partial solution, PS\textsuperscript{i}, is

\[
W_{1}(a,s_{1})W_{2}(t_{1},s_{2}) \cdots W_{i}(t_{i-1},s_{i})F(t_{i},? ,s_{i})V_{i}(t_{i},s_{i-1}) \cdots V_{i}(t_{i},?)
\]

where W\textsubscript{i} is the copy of the relation W obtained by applying the recursive rule r\textsubscript{r} the i\textsuperscript{th} time.

If the expression given above is evaluated from left to right (e.g. using the Single Wavefront Method), then there is no variable in common between adjacent W relations. As a result, cartesian products and therefore huge intermediate results will be obtained. On the other hand, the expression can be rewritten as

\[
W_{1}(a,s_{1})V_{2}(t_{2},s_{2})W_{3}(t_{3},s_{3})V_{4}(t_{4},s_{4}) \cdots F(t_{i},? ,s_{i-1}) \cdots V_{i}(t_{i},?)
\]

Thus, if the expression is evaluated from left to right, only joins are taken.

We now state a necessary and sufficient conditions for the relations W\textsubscript{i} and V\textsubscript{i+1} to have a joining attribute.

Definitions: The variables \{x\textsubscript{1}, \ldots , x\textsubscript{p}\} appeared in the head of a rule are distinguished. All other variables are non-distinguished. A position in a predicate of a rule is distinguished if it is occupied by a distinguished variable; otherwise, it is non-distinguished.

Consider predicate A in the recursive rule. For each distinguished variable in A, there is a position in the head of the rule containing the variable. The set of all such positions in R\textsubscript{h} are the A-related positions in R\textsubscript{a}. The same set of positions in the body, R\textsubscript{b}, are the A-related positions in R\textsubscript{b}.

Notation Var(A) is the set of variables in predicate A. Pos(R\textsubscript{b}, B) are the set of B-related positions in R\textsubscript{b}.

We will be using A and B to represent W or V.

Proposition 1: For every i \geq 1, for every k \geq i+1, in PS\textsuperscript{k}, base predicates A\textsubscript{i} and B\textsubscript{i+1} have at least a joining attribute iff in the recursive rule,

\[ \text{Var}(A) \cap \text{Var}(\text{Pos}(R_{b}, B)) \text{ is non-empty.} \]

Using the above example, \text{Var}(W) = \{x,s\}, the V-related position is \{3\}. Thus, \text{Var}(\text{Pos}(R_{b}, V)) = \{s\}. Thus, since s is in common, W\textsubscript{i} and V\textsubscript{i+1} have a common joining attribute.

Note: Proposition 1 applies to 4 combinations of V and W. It is easy to check whether V\textsubscript{i} and W\textsubscript{i} are joinable or not. A necessary and sufficient condition for V\textsubscript{i} and W\textsubscript{i} to be joinable is that \text{Var}(V) \cap \text{Var}(W) is non-empty.

A similar result can be obtained for the joinability between the W's and F and between the V's and F.

Proposition 2: For every k \geq 1, in Exp(k), base relations A\textsubscript{k} and F\textsubscript{k+1} are joinable iff \text{Var}(A) \cap \text{Var}(\text{Pos}(R_{b}, F)) is non-empty.
(ii) Excessive Joins

The preceding subsection emphasizes joining between adjacent predicates. However, excessive joining, in particular between non-adjacent predicates, can cause inefficiency.

**Example 2:** Consider the following pair of rules:

\[
R(x,y,z,t) :\leftarrow F(x,y,z,t) \\
R(x,y,z,t) :\leftarrow W(x,s)R(s,s,y,u)V(u,z,t)
\]

and the query \( R(a,?,?,?) \).

It is easy to verify that the 1\(^{\text{st}}\) partial solution, \( P_{1} \), can be written as:

\[
W_{1}(a,s_{1})W_{2}(s_{1},s_{2}) \cdots F(s_{i},s_{i-1},u_{i}) \cdots V_{3}(u_{i},u_{i+1}) \\
V_{2}(u_{2},u_{1})V_{1}(u_{1},?,?)
\]

Note that each predicate joins with its adjacent predicates so that an evaluation from left to right seems reasonable. However, after constructing the intermediate result for \( W_{1}W_{2} \cdots W_{j} \), it is necessary to save the attributes \( s_{k}, 3 \leq k \leq j \leq i-1 \), which are in common between \( W_{k} \) and \( V_{k+1} \); and \( s_{j} \), which is in common between \( W_{j} \) and \( W_{j+1} \). Thus, the number of attributes in the intermediate relation and therefore its size increases. This causes inefficiency.

The example given above illustrates that if \( W_{j} \) joins not only with its adjacent \( W \)’s (i.e. \( W_{j-1} \) and \( W_{j+1} \)) but also with non-adjacent \( V \)’s then the "excessive" joins will force us to save a large number of attributes for the intermediate results.

The following result states a necessary and sufficient condition that each \( A(W \) or \( V) \) does not join with non-adjacent \( W \)’s and \( V \)’s.

**Proposition 3:** For all \( i \) and all \( j, 3 \leq j \leq i \), in \( P_{i} \), \( A_{j} \) is not joinable with \( W_{m} \) nor with \( V_{m} \) for \( 1 \leq m \leq j-2 \) iff in the recursive rule, all positions in \( \text{Pos}(R_{b},A) \) are non-distinguished.

Similarly, a necessary and sufficient condition that the \( F \) does not join with non-adjacent \( W \)’s and \( V \)’s is given as follows.

**Definition:** A position in \( R \) is self-related if in \( r_{v} \), there is a variable appearing in that position in both \( R_{b} \) and \( R_{b} \).

**Proposition 4:** For all \( i, 2 \leq i \) in \( P_{i} \), \( F \) is not joinable with either \( W_{i} \) or \( V_{i} \), \( 1 \leq i \leq i-1 \), iff in \( r_{v} \), all distinguished positions in \( \text{Pos}(R_{b},F) \), if exist, are in \( \text{Pos}(R_{b},F) \cup \text{Pos}(R_{b},W) \) and are self-related.

Even if there is no joining between non-adjacent \( V \)’s and \( W \)’s, there could be joining between \( W_{k} \) and \( V_{k+1} \). This again may cause the saving of too many attributes, resulting in inefficient computation, as illustrated by the following example.

**Example 3:** Consider the following pair of rules:

\[
R(x,y,z) :\leftarrow F(x,y,z) \\
R(x,y,z) :\leftarrow W(x,s)R(s,s,y,u)V(y,z,t)
\]

and the query \( R(a,?,?,?) \).

It is easy to verify that the 1\(^{\text{st}}\) partial solution, \( P_{1} \), can be written as:

\[
W_{1}(a,s_{1})W_{2}(s_{1},s_{2})W_{3}(s_{2},s_{3},s_{4}) \cdots F(s_{i},u_{i}) \cdots V_{2}(u_{2},t_{1},u_{1})V_{1}(u_{1},?,?)
\]

Note that each predicate joins with its adjacent predicates so that an evaluation from left to right seems reasonable. However, after constructing the intermediate result for \( W_{1}W_{2} \cdots W_{j} \), it is necessary to save the attributes \( t_{k}, 1 \leq k \leq j \leq i-1 \), which are in common between \( W_{k} \) and \( V_{k+1} \), and \( s_{j} \), which is in common between \( W_{j} \) and \( W_{j+1} \). Thus, the number of attributes in the intermediate relation and therefore its size increases. This causes inefficiency.

If there is no joining between non-adjacent \( V \)’s and \( W \)’s (i.e. satisfying the conditions of Proposition 3), no joining between each \( V \) and its adjacent \( W \)’s (i.e. violating the conditions of Proposition 1), and no joining between each \( F \) and its non-adjacent \( V \)’s and \( W \)’s (satisfying the conditions of Proposition 4), then the number of attributes of the intermediate relations will not grow. The following example illustrates this idea.

**Example 4:** Consider the rules

\[
r_{e} : R(x,y) :\leftarrow F(x,y) \\
r_{r} : R(x,y) :\leftarrow W(x,u)R(u,v)V(y,v)
\]

\[
\text{Var}(\text{Pos}(R_{b},W)) = \{u\}, \text{Var}(\text{Pos}(R_{b},V)) = \{v\} \text{ and } \text{Var}(\text{Pos}(R_{b},F)) = \{u,v\}. \text{ Since } u \text{ and } v \text{ are non-distinguished, } V_{i} \text{ cannot join with non-adjacent } V \text{'s and } W \text{'s, } W_{i} \text{ cannot join with non-adjacent } V \text{'s and } W \text{'s, and } F \text{ cannot join with non-adjacent } V \text{'s and } W \text{'s. Furthermore, } \text{Var}(W) = \{x,u\} \text{ and } \text{Var}(V) = \{v,y\}. \text{ Since } \text{Var}(A) \cap \text{Var}(\text{Pos}(R_{b},B)) \text{ is empty for } A \neq B, \text{ A in } \{V,W\}, \text{ A}_{k} \text{ does not join with } \text{B}_{k+1} \text{ (by Proposition 1). Finally, } \text{Var}(W) \cap \text{Var}(V) \text{ is empty implies } W_{k} \text{ does not join with } V_{k}.
\]

The \( k \)\(^{\text{th}}\) partial solution, \( P_{k} \), is
\[ W_1(x_1,u_1)W_2(u_1,u_2) \cdots W_k(u_{k-1},u_k)V_{k+1}(u_k,\ldots) \]
\[ V_1(\ldots) \]

In computing the expression given above, after joining \( W_{i-1} \) with \( W_i \), the joining attribute \( u_{i-1} \) can be discarded. A similar situation applies to the \( V \)'s. Thus, the number of attributes of the intermediate relation will not grow. \( \square \)

(iii) Joining Attributes

In the Central Wavefront Method [HaLu], the \( i \)th partial solution, \( PS^i \), is expressed in terms of the \((i-1)\)th expression as follows.

\[ PS^i = W_1(W_2 \cdots W_{i-1}V_1 \cdots V_2)V_1 \]
\[ = W_1PS^{i-1}V_1 \quad (2) \]

Among all the wavefront methods, the Central Wavefront Method has the least number of joins. However, the equation given above, is not quite correct, as illustrated by the following example.

Example 5: Consider the rules

\[ r_0 : R(x,y,z,u,v) \rightarrow F(x,y,z,u,v) \]
\[ r_i : R(x,y,z,u,v) \rightarrow W(x,z,s)R(s,s,r,u,t)V(v,y,t,r) \]

Consider \( PS^2 \) and \( PS^3 \).

\[ PS^2 = W_1(x,z,s_1)W_2(s_1,y_1,s_2)F_3(s_2,s_2,s_1,u,t_1)W_2(t_1,v_1) \]
\[ PS^3 = W_1(x,z,s_1)W_2(s_1,y_1,s_2)W_3(s_2,s_1,s_3)F_4(s_3,s_3,s_2,u,t_2) \]

The joining attribute between \( W_1 \) and \( W_2 \) in \( PS^2 \) is \( s_1 \) at position 3 in \( W_1 \) and at position 1 in \( W_2 \). But the joining attributes between \( W_2 \) and \( W_3 \) in \( PS^2 \) are \( s_1 \) at position 1 in \( W_2 \) and at position 2 in \( W_3 \) and \( s_2 \) at position 3 in \( W_2 \) and at position 1 in \( W_3 \). Thus, \( W_2W_3 \) in \( PS^2 \) is not equal to \( W_2W_3 \) in \( PS^3 \). In other words, equation (2) is not correct, for \( i=3 \). \( \square \)

In order that equation (2) is correct, we need \( W_2 \cdots W_{i-1}V_1 \cdots V_2 = W_1 \cdots W_{i-2}V_{i-1} \cdots V_1 \). This is valid, if the joining attributes between \( W_i \) and \( W_{i-1} \), \( 3 \leq i \leq 5 \) are the same as those between \( W_2 \) and \( W_1 \); the joining attributes between \( V_j \) and \( V_{j-1} \) are the same as those between \( V_2 \) and \( V_1 \); the joining attributes between \( W_1 \) and \( F_{i+1} \) are the same as those between \( W_{i-1} \) and \( F_i \) and finally the joining attributes between \( F_i \) and \( V_{i-1} \) are the same as those between \( F_{i+1} \) and \( V_i \). The following result states that if each \( W \) does not join with non-adjacent \( W \)'s nor with non-adjacent \( V \)'s (which is the condition stated in Proposition 3), then the joining attributes between \( W_i \) and \( W_{j-1} \) and \( W_j \), \( 3 \leq j \leq i \). Proposition 5: For all \( k \) and all \( j \), \( 3 \leq j \leq i \), if in \( PS^k \), \( W_j \) is not joinable with \( W_i \) nor with \( W_{j-1} \), \( 1 \leq i \leq j-2 \), then the set of joining attributes of \( W_1 \) and \( W_2 \) is the same as that of \( W_{j-1} \) and \( W_j \).

A similar result is for the joining between the \( W \)'s and the \( F \)'s and between the \( V \)'s and the \( F \)'s.

Proposition 6: If \( F_{k+1} \) is not joinable with \( V_j \) nor with \( W_i \), \( 1 \leq i \leq k-1 \), in \( PS^k \), for all \( k \geq 2 \), then the set of joining attributes of \( W_1(V_j) \) and \( F_2 \) in \( PS^2 \) is the same as that of \( W_i(V_j) \) and \( F_{i+1} \) in \( PS^i \), for every \( i \geq 2 \).

Example 6: With the recursive rules given in Example 5, \( Pos(R,b,W) = \{1,3\} \); \( y \) at position 3 in \( R_b \) is distinguished. By Proposition 3, the hypothesis of Proposition 5 is violated and therefore equation (2) is not guaranteed.

Suppose the set of rules are

\[ r_0 : R(x,y,z,u,v) \rightarrow F(x,y,z,u,v) \]
\[ r_i : R(x,y,z,u,v) \rightarrow W(x,z,s)R(s,s,r,u,t)V(v,y,t,r) \]

\[ Pos(R,b,W) = \{1,3\}; \quad Pos(R,b,V) = \{2,5\} \quad \text{and} \quad Pos(R,b,F) = \{1,2,3,4,5\} \]

Since all the positions in \( R_b \) are non-distinguished, \( V \) and \( W \) will not join with non-adjacent \( V \)'s and \( W \)'s, by Proposition 3 and \( F \) will not join with non-adjacent \( V \)'s and \( W \)'s by Proposition 4. Since the hypothesis of Proposition 5 is satisfied, the set of joining attributes of \( W_1(V_1) \) and \( W_2(V_2) \) is the same as that of \( W_{j-1}(V_{j-1}) \) and \( W_j(V_j) \). A similar situation applies to the joining between the \( F \)'s and the \( V \)'s and \( W \)'s. Thus equation (2) now holds.

It is easy to verify that \( PS^k = \)

\[ W_1(x,z,s_1)W_2(s_1,y_1,s_2)W_3(s_2,s_2,s_1,u,t_1)W_2(t_1,v_1)F_{k+1}(s_3,s_3,s_2,u,t_2)V_1(v,y_1,v,\ldots) \]

Since, the joining conditions are as indicated in Proposition 5 and 6, the Central Wavefront Method is applicable. \( \square \)
5. Efficient Evaluation of Wavefront Methods:

In this section, we state the conditions that allow the wavefront methods to be evaluated efficiently. Even a single wavefront expression can be evaluated in many different ways. For example, in the Single Wavefront Method, PS can be written as

(I) \((W_1 W_2 \cdots W_i) F_{i+1} V_i \cdots V_1\)

or

(II) \((W_1 V_1 W_2 V_2 \cdots W_i V_i) F_{i+1}\)

or

(III) \((W_1 V_2 W_3 V_4 \cdots W_i V_i) F_{i+1} W_i V_{i-1} \cdots W_i V_i \) i even

\((W_1 V_2 W_3 \cdots W_i V_i) F_{i+1} V_i V_{i-1} \cdots V_1 \) i odd.

where the subexpressions within parenthesis are saved for the next iteration.

Because of the diversity in computing the expression in different ways using the wavefront methods and the limitation in space, we shall state the conditions for evaluating (I) efficiently using the Single Wavefront Method and those for evaluating the Central Wavefront Method.

**Proposition 7:**

Let Q be a recursive query. If the following conditions are satisfied, then Q can be evaluated efficiently using (I) of the Single Wavefront Method.

(i) Q initializes some variable \(\text{Var}(\text{Pos}(R_{h,F}))\) for some base predicate \(A\).

(ii) In the recursive rule \(r\), for each base predicate \(A\), all positions in \(\text{Pos}(R_{b,F})\) are non-distinguished, \(\text{Var}(V) \cap \text{Var}(W)\) is empty and \(\text{Var}(A) \cap \text{Var}(\text{Pos}(R_{b,F}))\) is empty, \(B \neq A\).

(iii) In \(r\), all distinguished positions in \(\text{Pos}(R_{b,F})\), if exist, are in \(\text{Pos}(R_{b,F}) \cup \text{Pos}(R_{b,W}) \cup \text{Pos}(R_{b,V})\) and are self-related.

(iv) In \(r\) for each base predicate \(A\) (\(V\) or \(W\)), \(\text{Var}(A) \cap \text{Var}(\text{Pos}(R_{b,F}))\) is not empty and \(\text{Var}(A) \cap \text{Var}(\text{Pos}(R_{b,A}))\) is not empty.

**Remarks:**

Condition (i) ensures that either \(V\) or \(W\) is instantiated so that the amount of data manipulated will not be excessive. If \(W\) is instantiated, then the formula should be evaluated starting from \(W_i\), as in (I); if \(V\) is instantiated, then the starting symbol should be \(V_i\), with the expression in (I) in reverse.

By Proposition 3, the first part of condition (ii) ensures that each \(V\) and each \(W\) will not join with non-adjacent \(V\)'s and \(W\)'s; the second part ensures that \(V_i\) and \(W_i\) do not join; the third part ensures that \(V_i(W_i)\) do not join with \(W_{i+1}\) and \(W_{i-1}(V_i+1\) and \(V_i-1\) by Proposition 1.

Condition (iii) ensures that \(F_{i+1}\) can possibly join with \(V_i\) and \(W_i\) only, by Proposition 4. This allows non-target attributes of \(F_{i+1}\) to be discarded after joining with \(V_i\).

Condition (iv) ensures that \(W_i\) and \(V_i\) join with \(F_{i+1}\), by Proposition 2. Furthermore, \(W_j\) joins with \(W_{j+1}\) and \(V_j\) joins with \(V_{j+1}\), by Proposition 1.

**Proposition 8:**

Let Q be a recursive query. If the following conditions are satisfied, then Q can be evaluated efficiently using the Central Wavefront Method.

(i) Q initializes some variables in \(\text{Var}(\text{Pos}(R_{h,F}))\) and some of which are at self-related positions.

(ii) In the recursive rule \(r\), all distinguished positions in \(\text{Pos}(R_{b,F})\), are in \(\text{Pos}(R_{b,F}) \cup \text{Pos}(R_{b,W}) \cup \text{Pos}(R_{b,V})\) and are self-related.

(iii) In \(r\), for each base predicate \(A\) (\(W\) or \(V\)), all positions in \(\text{Pos}(R_{b,A})\) are non-distinguished, \(\text{Var}(\text{Pos}(R_{b,F})) \cap \text{Var}(B)\) is empty, \(A \neq B\).

(iv) In \(r\) for each base predicate \(A\) (\(W\) or \(V\)), \(\text{Var}(A) \cap \text{Var}(\text{Pos}(R_{b,F}))\) is not empty and \(\text{Var}(A) \cap \text{Var}(\text{Pos}(R_{b,A}))\) is not empty.

The remarks are similar to those of Proposition 7.

6. Conclusion

In this paper, we characterize the properties of the expressions generated from the pair of rules which define linear recursive queries. These properties will be essential not only to allow efficient recursive query processing using the wavefront methods, but also to the understanding of the type of expressions generated by linear recursive rules.
References


[HaLu] Jiawei Han, and Hongjun Lu, "Some Performance Result on Recursive Query Processing in Relational Database Systems", IEEE Proc. of Inter. Conference on Data Engineering, 1986.


