In certain database applications such as deductive databases, batch query processing, and recursive query processing etc., usually a single query gets transformed into a set of closely related database queries. Also, great benefits can be obtained by executing a group of related queries all together in a single unified multi-plan instead of executing each query separately. In order to achieve this Multiple Query Optimization (MQO) identifies common task(s) (e.g. common subexpressions, joins, etc.) among a set of query plans and creates a single unified plan (multi-plan) which can be executed to obtain the required outputs for all queries at once. In this paper a new heuristic function \( h_c \), dynamic query ordering heuristics, and Depth-First Branch-and-Bound (DFBB) are defined and experimentally evaluated, and compared with existing methods which use \( A^* \) and static query ordering. Our experiments show that all three of \( h_c \), DFBB, and dynamic query ordering help to improve the performance of our MQO algorithm.

The objective of multiple query optimization (MQO) is to exploit the benefits of sharing common tasks in the access plans for a group of queries. In certain database applications, e.g. deductive query processing, batch query processing and recursive query processing, often a group of queries are submitted together to the DBMS for execution. The traditional approach of processing queries one at a time will be inefficient especially when there is a high number of queries sharing common relations and predicates. MQO identifies common sub-expressions among queries and creates an integrated query execution plan in which common tasks are evaluated only once.

The idea of processing multiple queries has been around for almost a decade (Chakravarthy, 1982; Finkelstein, 1982; Chakravarthy, 1986; Sellis, 1986; Sellis, 1988]. Grant et al (1982) used a depth-first based approach to the problem of common sub-expression analysis. Chakravarthy and Minker (1986), used an extended version of the query graph (Wong, 1976), called connection graph, to represent a set of queries. A query decomposition algorithm guided by a set of heuristics was used to evaluate all of the queries simultaneously. Chakravarthy et al (1988) addressed the MQO problem at various levels of detail, depending on the cost measure used. Sellis showed the MQO problem to be NP-hard (Sellis, 1990), and gave a state space search formulation (Sellis, 1986; Sellis, 1988). \( A^* \) is used as the search algorithm with bounding functions and intelligent state expansion, based on query ordering, to eliminate states of little promise rapidly. Subsequent improvement or variation of Sellis’s effort on the MQO problem has been reported in several papers (Park, 1988; Cosar, 1991; Lim, 1992). In (Cosar, 1991), the Sellis’ \( A^* \) algorithm is revised by having an improved heuristics function which prunes search space more effectively while still guaranteeing to find an optimal solution. Simulated annealing technique has also been experimentally analyzed to handle larger MQO problems that cannot be solved using \( A^* \) in a reasonable time with the currently available heuristics.

Our contributions

One of the fundamental parameters in Sellis’ work is the ordering of queries so as to decrease the error in heuristic cost.
calculation function and also to group together queries which have a high amount of shared task(s). There are several ordering heuristics used by Sellis which are computed only once at the beginning and remain constant throughout the search. However, we have observed that an initial ordering may become ineffective, since once a plan for a query is merged to the multi-plan, all of the sharings between the discarded alternative plan(s) and remaining queries become invalid.

To account for this shortcoming of the query ordering heuristics we have adopted a set of dynamic query ordering algorithms so that the order at which plans are merged to the multi-plan dynamically changes based on the current partial multi-plan to be augmented by a new plan. Experimental results show that significant gains are obtained by employing dynamic query ordering.

As a second contribution we have also analyzed Depth-First Branch-and-Bound (DFBB) as a new alternative algorithm to A* for solving MQO problems. The DFBB algorithm demonstrates some preferable characteristics over A* algorithm. Using A* algorithm as a baseline algorithm, we conducted several experiments to verify the advantages of DFBB in the MQO domain. Improvement in the performance of the A* algorithm and DFBB is dependent on (i) the heuristic function used for estimating the lower bound on the cost of a given path to an optimal plan, and (ii) a good query ordering. Sellis proposed a heuristic function (f) as well as some alternative query orderings for his A* algorithm [Sell90]. We prove that a better heuristic function (f) can be obtained and propose Successive Augmentation (Swami, 1988) as an efficient method to calculate the initial upper bound. Lastly, we experimentally show that the heuristics of selecting the plan (and thus the query) with the largest sharing with the current partial multi-plan can be used to adjust query ordering dynamically to prune the search space more effectively. Equipped with a better heuristic function, and the dynamic query ordering heuristics, DFBB is demonstrated to perform much better than the Sellis’ A* algorithm. The use of depth first search also helps to reduce the cost of calculating the heuristic function as it reduces the number of “plan merge” operations which is the operation used for adding a plan to the current multi-plan by considering the shared task(s).

**Paper outline**

Our paper is structured as follows. Section 2 gives a formal definition of the MQO problem. We examine the suitability of various search algorithms for MQO in section 3. In section 4, we present our new heuristic function (h') and show that it enables us to expand much less states than the previously proposed heuristic function (h) when applied to A* algorithm. A Successive Augmentation algorithm is introduced for deciding the initial upper bound in section 5. We then present the new dynamic query ordering heuristics in section 5.1. We ran our heuristics on an experimental set of query plans and compared the results with those obtained by Sellis’ query ordering heuristics which are presented using three tables in section 6. Our conclusions are presented in section 7.

**Problem Formulation**

In this section, we present a formulation of the MQO problem which is due to Sellis (Sellis, 1988; Sellis, 1990).

Let Q₁, ... , Qₙ be n queries to be optimized together. Query Qᵢ has a set of nᵢ alternative plans for its evaluation, namely Pᵢ,1, Pᵢ,2, ..., Pᵢ,nᵢ.

Plan Pᵢ,j is a set of tasks {tᵢ,1, tᵢ,2, ..., tᵢ,nᵢ,j}. A task tᵢ,j has an associated cost of cost(tᵢ,j).xx

A solution, S, to the MQO problem is a set of plans Pₛ = {P₁,1, P₁,2, ..., Pₙ,nᵢ}.

Let Tᵢ = ∪(1≤j≤nᵢ,j) Pᵢ,j be the set of tasks in the solution S. Now, cost(S) = Σ(cost(tᵢ,j)) tᵢ,j ∈ Tᵢ is the cost of the solution.

An optimal solution S* is such that cost(S*) is minimal.

Example 1 shows a sample MQO problem with two queries and five plans. Query Q₁ has plan P₁,1 and P₁,2 while query Q₂ has plans P₂,1, P₂,2 and P₂,3. A plan is made up of a set of tasks, each with a positive cost.

**Example 1:**

Let the plans for Q₁ and Q₂ have the following task sets:

| P₁,1 = {t₁, t₂, t₃}; | P₁,2 = {t₄, t₅}; |
| P₂,1 = {t₁, t₆, t₇}; | P₂,2 = {t₂, t₈, t₉}; | P₂,3 = {t₅, t₁₀}; |

The task costs are:

<table>
<thead>
<tr>
<th>t₁</th>
<th>t₂</th>
<th>t₃</th>
<th>t₄</th>
<th>t₅</th>
<th>t₆</th>
<th>t₇</th>
<th>t₈</th>
<th>t₉</th>
<th>t₅, t₁₀</th>
</tr>
</thead>
<tbody>
<tr>
<td>cost</td>
<td>40</td>
<td>30</td>
<td>5</td>
<td>35</td>
<td>20</td>
<td>10</td>
<td>5</td>
<td>10</td>
<td>30</td>
</tr>
</tbody>
</table>

Six solutions are possible, with the following costs:

| cost(S(P₁,1, P₂,1)) = cost(t₁) + cost(t₄) + cost(t₅) + cost(t₆) + cost(t₇) = 90 |
| cost(S(P₁,2, P₂,1)) = cost(t₄) + cost(t₅) + cost(t₆) + cost(t₇) + cost(t₈) = 90 |
| cost(S(P₁,1, P₂,2)) = cost(t₁) + cost(t₂) + cost(t₃) + cost(t₄) + cost(t₅) = 125 |
| cost(S(P₁,1, P₂,3)) = cost(t₁) + cost(t₂) + cost(t₃) + cost(t₄) + cost(t₅) = 110 |
| cost(S(P₁,2, P₂,2)) = cost(t₄) + cost(t₅) + cost(t₆) + cost(t₇) + cost(t₈) = 110 |
| cost(S(P₁,2, P₂,3)) = cost(t₄) + cost(t₅) + cost(t₆) + cost(t₇) + cost(t₈) = 85 |

The minimum cost plans for the queries Q₁ and Q₂ are P₁,2 and P₂,3, with costs 55 and 45, respectively. The minimum cost multi-plan, however, is {P₁,1, P₂,3}. It is important to note that an optimal multi-plan is not necessarily made up of the individual optimal plans for each query. Similarly, it is not always necessary that an optimal multi-plan will include a plan.
Related Content

An Alternative Fit through Problem Representation in Cognitive Fit Theory
Hock Chuan Chan, Suparna Goswami and Hee-Woong Kim (2012). Journal of Database Management (pp. 22-43).
[www.igi-global.com/article/alternative-fit-through-problem-representation/65540?camid=4v1a](www.igi-global.com/article/alternative-fit-through-problem-representation/65540?camid=4v1a)

Transformations Between UML Diagrams
[www.igi-global.com/article/transformations-between-uml-diagrams/3298?camid=4v1a](www.igi-global.com/article/transformations-between-uml-diagrams/3298?camid=4v1a)

A Database Interface for Link Analysis
[www.igi-global.com/article/database-interface-link-analysis/3327?camid=4v1a](www.igi-global.com/article/database-interface-link-analysis/3327?camid=4v1a)

INCA's: Managing Dynamic Workflows in Distributed Environments
[www.igi-global.com/article/incas-managing-dynamic-workflows-distributed/51158?camid=4v1a](www.igi-global.com/article/incas-managing-dynamic-workflows-distributed/51158?camid=4v1a)