Parallel Query Processing and Edge Ranking of Graphs

Dariusz Dereniowski *, Marek Kubale **

Department of Algorithms and System Modeling,
Gdańsk University of Technology, Poland,
{deren,kubale}@eti.pg.gda.pl

Abstract. In this paper we deal with the problem of finding an optimal query execution plan in database systems. We improve the analysis of a polynomial-time approximation algorithm due to Makino et al. for designing query execution plans with almost optimal number of parallel steps. This algorithm is based on the concept of edge ranking of graphs. We use a new upper bound for the edge ranking number of a tree to derive a better approximation ratio for this algorithm. We also present some experimental results obtained during the tests of the algorithm on random graphs in order to compare the quality of both approximation ratios on average. Both theoretical analysis and experimental results indicate the superiority of our approach.

1 Introduction

A parallel query execution is a way for improving performance in large database systems. In general, there are three types of parallelism in database systems: *inter-query parallelism*, *inter-operator parallelism* and *intra-operator parallelism*. In all cases the goal is processor allocation but at different levels. In inter-query parallelism we have a set of queries (some of them can be executed concurrently) and we have to find their parallel scheduling. If more than one processor is assigned to a query then the two other mentioned types of parallelism can be exploited as well. In inter-operator parallelism we schedule operations within a single query. In this paper we consider joins only as they are the most time consuming operations. Thus, we schedule join operations by assigning processors to joins and determining which operations can be computed independently. In the last type of parallel scheduling, intra-operator parallelism, we split one join operation among several processors in order to compute it efficiently. In this paper we present results which can be applied in inter-operator parallelism.

In the inter-operator parallelism we have two main problems to consider: join sequence selection and allocation of processors to joins. In this paper we deal

* Supported in part by KBN grant 3 T11C 01127
** Supported in part by KBN grant 4 T11C 04725

with the first issue. Furthermore, the algorithm under consideration does not deal with the join execution costs. To avoid large intermediate relations we can use a technique described in [8], where the graph ranking approach can still be used.

Finding an optimal parallel *query execution plan* (QEP) is NP-hard. Thus, for complex queries the problem is intractable and most approaches are based on heuristics which generally have no formal analysis (such as worst case bounds) on their optimality [12]. Makino, Uno and Ibaraki [9] gave an algorithm with a worst case performance guarantee for solving this problem. In the sequel we will call their approach as the *MUI algorithm*. In the next section we describe the concept of edge ranking and the MUI algorithm. In Section 3 we improve on the prior analysis of MUI. Finally, Section 4 gives some experimental results.

2 Preliminaries

The *join graph* of a query is defined as a simple graph G such that the vertices of G correspond to the base relations and two vertices are adjacent in G if the corresponding relations have to be joined during the execution of a query [4, 8, 11]. In this paper we assume that the join operations are executed in steps. We are interested in finding parallel scheduling of the joins, so we assume that in each step many join operations can be performed. If a base relation is used in two different join operations (which means that in the query graph the edges which correspond to these operations are incident to a common vertex) then these joins cannot be computed in the same step. The scheduling of the join operations can be represented as a rooted tree T', where nodes of this tree represent the join operations and a particular join can be evaluated only if all its descendants have been already computed. Furthermore, if two nodes are unrelated in T' then the corresponding joins can be computed in parallel. The above data structure is known as the *operator tree*. For more detailed description of operator trees see e.g. [1, 5].

If an operator tree T' has been computed the join can be scheduled in such a way that the tree is processed in a bottom-up fashion – in the ith step all joins which correspond to the nodes in the ith level of T' are computed, where the last level is the root of T'. Note that the height of T' (the length of the longest path from the root to a leaf) is the number of parallel steps required to process the query. So, it is desirable to find a tree T' of small height. Note that if some relations x and y have been joined in the ith step (let z be the resulting intermediate relation) then in all joins in the later steps we use z instead of x and y.

In the following we assume that a join graph $G = (V(G), E(G))$ is given, where $|V(G)| = n$ and $|E(G)| = m(G) = m$. We denote by $\Delta(G)$ the maximum vertex degree of G, in short the *degree* of G. An *edge k-ranking* of G is a function $c : E(G) \to \{1, \ldots, k\}$ such that each path between edges x, y satisfying $c(x) = c(y)$ contains an edge with a greater color. The smallest integer k such that there exists an edge k-ranking of G is the *edge ranking number* of G and is denoted

by $\chi'_r(G)$. Makino et al. [9] introduced the *minimum edge ranking spanning tree* (MERST) problem. In this problem the goal is to find a spanning tree whose ranking number is as small as possible.

Spanning trees and their edge rankings can be used to find good QEPs (which in our case is equivalent to finding low height separator trees) in the following way. In order to complete the execution of the query, we have to compute joins which correspond to edges forming a spanning tree in the query graph G. Indeed, if C is a cycle with l edges in G then computing any $l-1$ joins from that cycle results in an intermediate relation obtained by joining all base relations corresponding to the vertices of C. Furthermore, if we have selected some spanning tree T of G, then the task is to find an optimal parallel scheduling of joins corresponding to the edges of T. We find an edge ranking c of T and schedule the operations in such a way that for any two edges $e_1, e_2 \in E(T)$ the join corresponding to e_1 is the father of the join corresponding to e_2 in an operator tree T' if and only if $c(e_1) > c(e_2)$ and every path connecting e_2 to an edge e with $c(e) > c(e_1)$ passes through the edge e_1. So, the number of colors used by c is the height of the operator tree T' obtained in this way. Furthermore, this approach is optimal in the sense that if the spanning tree of the query graph is given then the height of an operator tree of minimum height equals to the edge ranking number of T [9].

Let us illustrate the above concepts in the following example. Consider a database query of the form

$$\begin{aligned} &\text{Select * from } A, B, C, D, E, F \text{ where } A.x = C.x \text{ and } A.y = D.y \text{ and} \\ &\qquad A.z = E.z \text{ and } A.t = B.t \text{ and } B.u = E.u \text{ and } B.v = F.v \ldots \end{aligned} \tag{1}$$

Fig. 1(a) depicts the corresponding query graph G containing edges $\{A, B\}$, $\{A, C\}$, $\{A, D\}$, $\{A, E\}$, $\{B, E\}$, $\{B, F\}$ which correspond to the join operations that have to be computed in order to complete the above query. The edges

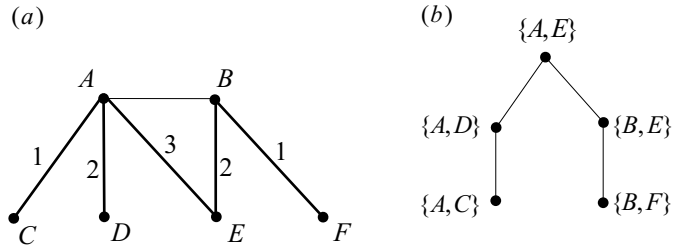

Fig. 1. (a) a query graph G, its spanning tree T and an edge ranking of T; (b) the corresponding operator tree T'

of a spanning tree T of G are marked with heavy lines in Fig. 1(a). Note that the degree of T is minimum over degrees of all spanning trees of G. Furthermore,

this is an optimal solution to the MERST problem for the graph G. We compute an edge ranking of T in order to create an operator tree. An example of such a coloring is given in Fig. 1(a) (the numbers labeling the edges of T are the colors). The largest color used is equal to 3. Thus, this query can be computed in parallel in three steps. The operator tree T' created on the basis of this edge ranking is given in Fig. 1(b). Nodes of T' are labeled with the corresponding join operations. Finally, the operator tree can be used to create a parallel schedule as described above. In this case the join operations $\{A, C\}$ and $\{B, F\}$ are computed independently in the first step. In the second step we have operations $\{A, D\}$ and $\{B, E\}$. However, instead of using the relation A (B) we use the intermediate relation $\{A, C\}$ ($\{B, F\}$) created previously. In the final step we join the intermediate relations corresponding to the sons of the root. The first relation is the result of joining A, C, D and the other was obtained by joining B, E and F.

Though the MERST problem is polynomially solvable for threshold graphs [8], it is NP-hard for general graphs [9]. As we described above, the difficulty lies in finding a required spanning tree of the query graph. The MUI algorithm solves MERST for general graphs with sublinear approximation ratio

$$\frac{\min\{(\Delta^* - 1)\log n/\Delta^*, \Delta^* - 1\}}{\log(\Delta^* + 1) - 1},$$

where Δ^* is the degree of a spanning tree whose Δ is as small as possible, and log stands for $\log_2$ [9].

The algorithm MUI can be described as follows:

1. Find a spanning tree T of G with $\Delta(T) \leq \Delta^* + 1$.
2. Find an optimal edge ranking of T.

In the first step we use a 1-absolute approximation polynomial-time algorithm given in [3]. The problem of finding an optimal edge ranking of a tree is polynomially solvable [6].

3 Worst-Case Analysis of MERST

The following upper bounds for the edge ranking number of a tree are used to derive the corresponding approximation ratios of MUI. In the following $\Delta = \Delta(T)$.

Theorem 1. ([9]) *If T is a tree with degree equal to Δ then*

$$\chi'_r(T) = \lceil \log n \rceil, \quad \text{if } \Delta = 0, 1, 2$$

$$\chi'_r(T) \leq \frac{(\Delta - 2)\log n}{\log \Delta - 1} = B_1(T), \quad \text{if } \Delta > 2.$$

Theorem 2. ([2]) *If T is a tree with degree equal to $\Delta > 1$ then*

$$\chi'_r(T) \leq \Delta \log_\Delta m = B_2(T).$$

The following simple lower bounds are also used

Lemma 1. ([9]) *For any tree T, $\chi'_r(T) \geq \max\{\Delta, \lceil \log n \rceil\}$.*

If T_o is an optimal solution to MERST for G, and T is a tree produced in the first step of algorithm MUI, then on the basis of Lemma 1 and Theorem 1 we have [9]

$$\begin{aligned} \frac{\chi'_r(T)}{\chi'_r(T_o)} &\leq \frac{(\Delta^* - 1)\log n/(\log(\Delta^* + 1) - 1)}{\max\{\Delta^*, \lceil \log n \rceil\}} \\ &\leq \frac{\min\{(\Delta^* - 1)\log n/\Delta^*, \Delta^* - 1\}}{\log(\Delta^* + 1) - 1} = R_1(G). \end{aligned} \tag{2}$$

Our approximation ratio for MUI, obtained on the basis of Lemma 1 and Theorem 2 is

$$\frac{\chi'_r(T)}{\chi'_r(T_o)} \leq \frac{(\Delta^* + 1)\log_{\Delta^*+1}(n-1)}{\max\{\Delta^*, \lceil \log n \rceil\}} = R_2(G). \tag{3}$$

Since $B_1 - B_2 = \Theta(\log n)$ for $\Delta \geq 4$ [2], by (2) and (3) we have $R_2(G) \leq R_1(G)$ for $\Delta^* \geq 3$. Note, that for a fixed value of Δ^* we have $R_1 = \Theta(\log n)$ and $R_2 = \Theta(1)$.

4 Experimental Results

Below we present some experimental results gained while testing MUI on random graphs. Each chart shows the values of parameters as functions of n, the size of graph. We generated graphs for each $n \in \{50, 60, \ldots, 150\}$. Each point denoted in the chart is the average of 100 values. All graphs created during the tests were connected. Note that if G is not connected then solving the MERST problem for G reduces to solving the problem for all connected components of G separately. Then a solution for G is a spanning forest and its edge ranking number is the maximum over the edge ranking numbers of the trees forming the forest. In order to create a random graph we constructed its random spanning tree and then, according to the graph density g, more edges were added. This means that for small g (i.e. $g = 0.005$) the random graphs were trees. Fig. 2 shows the results of the computer experiments.

Fig. 2(a) depicts the relation between the edge ranking number of a tree and the bounds B_1 and B_2. The degree of all trees generated was equal to 10. Two types of inaccuracies are included in the approximation ratios considered in this paper. The first follows from the fact that the MUI algorithm uses a heuristic for finding a minimum degree spanning tree. However, it is possible

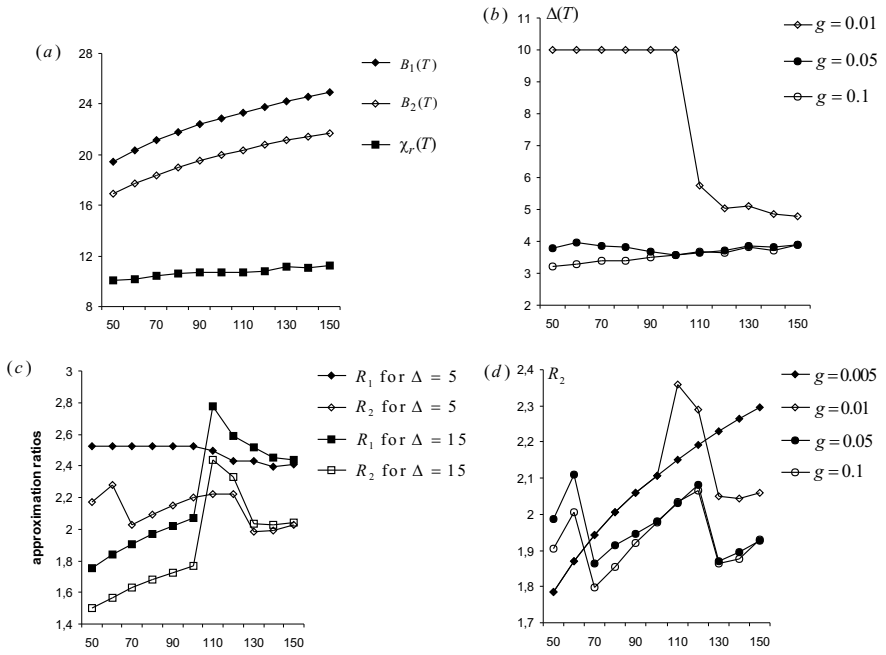

Fig. 2. Experimental results

to create examples of graphs for which a spanning tree with minimum vertex degree is not the one with minimum edge ranking number. Then, there is some inaccuracy in bounding the edge ranking number when only n and Δ are given. Thus, the purpose of comparing the edge ranking number with its bounds is that the quality of the approximation ratio directly depends on the bounds used.

The second test is presented in Fig. 2(b), where we tested only the first part of MUI, i.e. we generated random graphs G with $\Delta(G) = 10$ and densities $g = 0.01, 0.05, 0.1$, respectively. We used only small values of g, because as the experiments show, already for these values of g, the degree of the spanning trees is very small. From the theoretical analysis it follows that once we obtain a spanning tree which is a path, the problem has been solved optimally, so we want to avoid such cases. Some values of the first function for $g = 0.05$ are equal to 10, because random graphs obtained in these cases were trees.

Fig. 2(c) compares approximation ratios R_1 and R_2. In this test the graph density $g = 0.01$. We used a small value of g, because as the previous experiment shows, for larger g the average over degrees of T was smaller than 4 and in that case we would rather use the theoretical analysis from Section 3 to compare R_1 and R_2.

Fig. 2(d) presents the approximation ratio R_2. In this test, for each value of n and g we generated random graphs G with $\Delta(G) = 10$. Then, we computed

R_2 for the spanning tree obtained in the first part of MUI. As before, each point is the average of 100 values (i.e. 100 different random graphs G were created). The first function, because of the small value of g, presents R_2 for trees with $\Delta = 10$. For smaller values of n and $g = 0.01$ graphs G are also trees. Larger graphs G are not acyclic. The approximation ratio R_2 is clearly not monotonic in Δ^*, which explains why the second function presented in Fig. 2(d) can get smaller and bigger values than the first one.

5 Summary

In Section 3 we gave a theoretical analysis which indicates that the new approximation ratio better describes the worst case behavior of the MUI algorithm. On then other hand, the experimental results presented in Section 4 compare both bounds in the average case. The mean value of R_1 (taken over all test cases) was 15% bigger than the mean value of R_2.

References

1. S. Chaudhuri, An overview of query optimization in relational systems, *Proc. PODS* (1998), Seattle (WA), USA.
2. D. Dereniowski, M. Kubale, Efficient parallel query processing by graph ranking, *Fundamenta Informaticae* (submitted).
3. M. Furer, B. Raghavachari, Approximating the minimum-degree Steiner tree to within one of optimal, *J. Algorithms* **17** (1994) 409-423
4. T. Ibaraki, T. Kameda, On the optimal nesting order for computing N-relational joins, *ACM Transactions on Database Systems* **9** (1984) 482-502.
5. M. Kremer, J. Gryz, A survey of query optimization in parallel databases, *Technical Report* CS-1999-04 York University (1999).
6. T.W. Lam, F.L. Yue, Optimal edge ranking of trees in linear time, *Proc. of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms* (1998) 436-445.
7. H. Lu, M.-C. Shan, K.-L. Tan, Optimization of multi-way join queries for parallel execution, *Proc. of the 17th Conference on Very Large Data Bases*, Barcelona, Spain, September 1991.
8. K. Makino, Y, Uno, T. Ibaraki, Minimum edge ranking spanning trees of threshold graphs, *LNCS* **2518** (2002) 428-440.
9. K. Makino, Y, Uno, T. Ibaraki, On minimum edge ranking spanning trees, *J. Algorithms* **38** (2001) 411-437.
10. P. de la Torre, R. Greenlaw, A.A. Schaffer, Optimal edge ranking of trees in polynomial time, *Algorithmica* **13** (1995) 529-618.
11. J. D. Ullman, *Principles of Database and Knowledge-Base Systems*, Vol. 1. Computer Science Press, Maryland, 1990.
12. P.S. Yu, M.-S. Chen, J. L. Wolf, J. Turek, Parallel query processing In: N.R. Adam, B.K. Bhargava, editors, *Advanced Database Systems*, *LNCS* **759** Springer-Verlag, 1993.
13. X. Zhou, M.A. Kashem, T. Nishizeki, Generalized edge-rankings of trees, *LNCS* **1197** (1997) 390-404.