Approximate Solutions for Pipelined Operator Tree Scheduling in Uniform Processors

Arash Termehchy and Mohammad Ghodsi
Computer Engineering Department
Sharif University of Technology, Tehran, Iran
Email: Termechi@ce.shrif.ac.ir, ghodsi@sharif.edu

ABSTRACT
Pipelined Operator Tree (POT) scheduling is an important problem in the area of parallel query optimization. A POT is a tree whose nodes represent query operators that can be run in parallel and edges represent communication between adjacent operators. The problem is to find a schedule for the POT that minimizes the total response time. This problem has only been previously addressed for homogeneous environments, but the new parallel database systems tend to be more heterogeneous. In this paper, we consider processors with different fixed speeds (called uniform processor system). This problem has been shown to be NP-hard even for identical processors. We propose three approximate algorithms for some special cases of the problem that provides good low performance ratio (or approximation factor) bound in the worst case. We will show that the performance ratios of these algorithms, if used for homogeneous systems, are lower than the previous results. For the general case, we propose an algorithm which has a constant bound on the performance ratio in the worst case.

KEY WORDS
parallel scheduling, parallel databases, approximate algorithms

1. Introduction

With emerging sophisticated applications on parallel database systems, such as decision support systems and data mining the need to minimize the query response time is more than ever. To reduce the complexity of this problem, some researchers have used a two-phase approach [13, 10, 7]: join ordering and query rewriting followed by parallelization and scheduling. In the second phase, atomic units of the query for parallel execution, called operators, are extracted first and then scheduled to provide the minimum response time. One of the most important issues that must be considered is the parallelism-communication trade-off [5, 6]. To consider this, the query to be scheduled is represented as a weighted operator tree in which each node represents an operator and each edge represents the timing constraints between operators [13, 11]. A timing constraint is either a precedence or parallel constraint. The parallel constraint requires that the two adjacent nodes behave as a producer-consumer system where the producer sends a long stream of communication data to the consumer. These two nodes, therefore, start and terminate their works approximately at the same time.

A weighted operator tree in which all edges represent parallel constraints is called a Pipelined Operator Tree (POT) [11]. The communication cost in the POT makes its scheduling different from the classical scheduling problems. Since that data is transmitted in long streams, the important aspect of communication cost is the CPU overhead of sending/receiving messages and not the delay for signal propagation. Communication cost is saved if adjacent nodes are assigned to one processor but this would decrease the degree of parallelism. Load of a processor in a schedule is the sum of the weights of the operators assigned to it plus the weight of the edges that connect nodes on this processor to the nodes on other processors. This optimization problem can also viewed as to find a schedule that minimizes the maximum load of the processors.

POT scheduling problem was first introduced by Hasan and Motwani for identical processor systems and was shown to be NP-hard [11]. They proposed several approximation algorithms for restricted cases of the POT. Chekuri *et. al.* devised two algorithms for the general case [4]. The first one, called LOCALCUTS, has the worst case performance ratio of $\frac{3+\sqrt{17}}{2}$ and runs in $O(n \log n)$ time. The second, called BOUNDEDCUTS, has a smaller performance ratio of $(1+\epsilon)2.87$, at the expense of higher time complexity of $O(\frac{1}{\epsilon} n \log n)$. Chekuri has also shown that there exists a polynomial time approximation schema (PTAS) for this problem [3]. This result is not practical because of its very high time complexity.

Efficient approximate solution for parallel query optimization in heterogeneous processor systems has been considered to be a challenging task [7, 10]. Systems based on network of workstations or PC clusters are good examples of such systems [1, 2, 14, 15]. The workstations may differ in processor speed, amount of memory and the speed and the number of attached disks. A practical example is given in [15] where a PC cluster based parallel database was used with heterogeneous clusters. Moreover, the commodity interconnections make pipelining as a practical way and sometimes the only way to speed up query execution [14, 2].

To the best of our knowledge, heterogeneity of resources has not been considered before in the context of the problem that we are dealing with. In this paper, we define POT scheduling on uniform processor system [9]. First, we extend LOCALCUTS and BOUNDEDCUTS algorithms for two uniform processors with lower worst case performance ratio compared to the homogeneous case. We then extend LOCALCUTS heuristics for a more frequently occurring case of the problem (which is described later) with the worst case performance ratio lower than the homogeneous case. We also propose an algorithm based on the LOCALCUTS heuristics to solve the problem in the general case and prove that the algorithm has a constant bound on its performance ratio.

The organization of this paper is as follows. An overview of the model and problem definition are discussed in section 2. In section 3, we present our POT scheduling algorithms for a system with two uniform processors. The solutions for POT scheduling on an arbitrary number uniform processors are presented in section 4.

2. The model and problem definition

The following definitions are based on earlier models presented in [11, 4].

A POT is represented as a weighted operator tree $\mathcal{P} = (V, E)$. The weight p_k of the node k is the time to run the operator in isolation assuming all communications are local. The weight c_{kj} of the edge from node k to node j is the additional CPU overhead that both k and j will incur for inter-operation communication if they are scheduled on different processors. A schedule of $\mathcal{P}$ on p processors is a partition of V, the set of nodes, into p sets $F_1, \cdots, F_p$ such that set F_k is assigned to processor k. The load of processor k, or L_k, is the cost of executing all nodes in F_k plus the overhead for communicating with nodes on other processors. That is, $L_k = \sum_{j \in F_k} [p_j + \sum_{l \notin F_k} c_{jl}]$. L_{max} is $\max_{1 \leq k \leq p} L_k$.

Two operations are used to modify the POT: $Collapse(k, j)$ is to replace adjacent nodes k and j by a single node k' having weight of $t_{k'} = t_k + t_j$. Edges connected to either k or j are connected to k' instead. Operation $Cut(k, j)$ is to delete edge e_{kj} and add its weight to those of node k and j. For example, in figure 1 the operations $Collapse(a, c)$, $Collapse(c, d)$, $Cut(a, b)$, and $Cut(c, e)$ have been performed. Collapse and cut operations should be interpreted as decisions to allocate nodes to the same or distinct processors respectively. An edge e_{kj} is called *worthless* if and only if $c_{kj} \geq p_k + \sum_{l \neq j} c_{kl}$ or $c_{kj} \geq p_j + \sum_{l \neq k} c_{jl}$. As shown in [11] for homogeneous case, we can convert each POT into a POT with no worthless edges, called monotone tree, by collapsing all its worthless edges. We then schedule the monotone tree. In a monotone tree, we use the following notations: $R_k = p_k + \sum_{j \in V} c_{kj}$, $R = \max R_k$, and $W = \sum_{k \in V} p_k$.

Our algorithms are based on two-stage approach [4] in which the following phases are performed: fragmentation, and the actual scheduling. In the fragmentation phase, we partition the tree into connected fragments by cutting some edges. Any edge that is not cut is collapsed. Fragmentation produces a set of fragments that are ready to be scheduled independently. The scheduling phase assigns the fragments produced by the first phase. Let M_k be cost of a fragment F_k. Then, $M_k = \sum_{j \in F_k} [p_j + \sum_{l \notin F_k} c_{jl}]$. The weight of the heaviest fragment is denoted by M. Clearly, based on definition of the monotone tree, R is a lower bound for M. C is the total communication cost incurred in a fragmentation which is twice the sum of the weights of the cut edges. So the total load L is $W + C$. In the algorithm presented, we use superscript $\star$ to denote the quantities in the optimal scheduling.

The problem that we solve is defined as follows. In a uniform processor system the processors $m_1, \cdots,$ and m_p have relative speeds of $s_1 \geq s_2 \geq \cdots \geq s_p$ respectively. POT scheduling on a uniform processor system is to find a schedule with minimum response time of $T = \max_{1 \leq k \leq p} \frac{L_k}{s_k}$. Obviously, POT scheduling on uniform processors is also NP-hard. It is easily shown that we can collapse the worthless edges in the uniform processor case as well.

3. Two uniform processor scheduling

We use the same two-stage approach. For scheduling, we use LPT algorithm as has been used for the identical (homogeneous) case. LPT algorithm for the uniform processor system [9] assigns tasks to processors in the decreasing order of their processing times. A task is taken from this list and is assigned to a processor whose finishing time is the earliest. We first find a relationship between fragmentation phase and the LPT schedule.

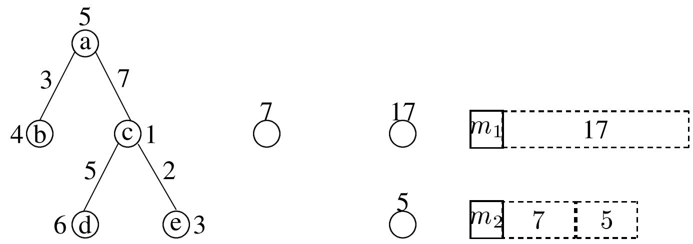
Clearly, heavy fragments increase load of processors and light fragments increase the communication cost of the edges cut. Therefore, a trade-off should be found for these effects.

Lemma 1 *Consider a fragmentation with $M \leq k_1 L^{\star}_{max}$ and $L \leq k_2 L^{\star}$ (k_1 and k_2 are real values ≥ 1), scheduling the produced fragments by LPT on two uniform processors yields a schedule with $T/T^{\star} \leq \max(k_1, 1.5k_2)$.*

Proof: Let m_l denote the processor that finishes last, and M_k be the lightest fragment assigned to m_l. Since we use LPT, M_k must be the last fragment assigned to m_l. We can remove all fragments which is lighter than M_k from the produced schedule without any change in T. Since $T^{\star}$ is fixed, there is no change in the performance ratio. Clearly, the total load of the new set of fragments $L^{'}$, is less than or equal to L, and the weight of the largest fragment M remains unchanged. For different values of k we have,

1. if $k \geq 2$, from definition of LPT we know that when scheduling M_k, we have

$$\forall 1 \leq j \leq 2 \,:\, T \leq \frac{L^{'}_j + M_k}{s_j},$$



(a)The original POT (b) Fragmentation (c) Scheduling

Figure 1. The two-stage approach.

which results in

$$T(\sum_{j=1}^{2} s_j) \leq \sum_{j=1}^{2} L_j^{'} + 2M_k,$$

in which $L_j^{'}$ is the load of m_j, when we schedule M_k. Since $\sum_{j=1}^{2} L_j^{'} = L^{'} - M_k$ and $L^{'} \leq L \leq k_2 L^{\star}$, we have,

$$T \leq k_2 L^{\star} / \sum_{j=1}^{2} s_j + M_k / \sum_{j=1}^{2} s_j.$$

Because $L^{\star}/\sum_{j=1}^{2} s_j \leq T^{\star}$, we get $T \leq k_2 T^{\star} + \frac{M_k}{L^{\star}} T^{\star}$. And thus $T \leq k_2 T^{\star} + \frac{k_2 M_k}{L^{'}} T^{\star}$. Since M_k is the lightest task in the schedule, we have $L^{'} \geq k M_k$. Therefore, $T \leq k_2 T^{\star} + \frac{k_2}{k} T^{\star}$. Thus,

$$T \leq 1.5 k_2 T^{\star} \leq \max(k_1, 1.5 k_2) T^{\star}$$

.

2. If $k = 1$, clearly M_k will be assigned to the processor with maximum speed. Therefore, $T \leq \frac{M_k}{s_2}$. From assumption of lemma, we conclude that

$$T \leq k_1 L_{max}^{\star} / s_2 \leq k_1 T^{\star} \leq \max(k_1, 1.5 k_2) T^{\star}.$$

□

We thus need to find algorithms for fragmentation the POT with minimum values of k_1 and k_2.

We modify LOCALCUTS algorithm based on our analysis for two uniform processor case. LOCALCUTS repeatedly picks up a leaf and determines whether to cut or collapse the edge from the leaf to its parent. It selects proper operation based on the ratio of the leaf weight to the weight of the edge to its parent. If the ratio is greater than an input parameter $\alpha > 1$, it will cut the edge, since this operation does not considerably increase the weight of the resulting fragments. If the ratio is less than α, the leaf is collapsed to the parent node. This is because the weight of the parent node will not increase substantially. In the algorithm, a *mother node* is defined as a node that all its children are leaves. In the algorithm that follows, we choose a proper value for α based on theorem 1.

Algorithm Local Cuts.
while there exists a mother node m with child j **do**
  if $p_j > \alpha c_{jm}$
   then Cut(j,m)
   else Collapse(j,m)
end while

Theorem 1 *The worst case performance ratio of* LOCALCUTS *followed by* LPT *for scheduling on two uniform processors is* 3.

Proof: We have $C \leq 2/(\alpha - 1)W$ in LOCALCUTS algorithm [4]. Therefore, $L = W + C \leq \frac{\alpha+1}{\alpha-1} W$. Since $W \leq L^{\star}$, then $L \leq \frac{\alpha+1}{\alpha-1} L^{\star}$. We also have $M < \alpha R$ for LOCALCUTS fragmentation [4]. Using lemma 1, we have $\frac{T}{T^{\star}} \leq \max(\alpha, 1.5\frac{\alpha+1}{\alpha-1})$

Because $1.5\frac{\alpha+1}{\alpha-1}$ is strictly decreasing in α and α is itself strictly increasing in α, minimizing the right hand side of the above inequality gives us $\alpha = 1.5\frac{\alpha+1}{\alpha-1}$, which is $\alpha = 3$. Thus, the worst case performance ratio of the algorithm is also equal to 3. □

The second algorithm that we modify is BOUNDEDCUTS. If R is small compared to $M^{\star}$, LOCALCUTS may cut expensive edges needlessly (maximum weight of fragments produced by LOCALCUTS is bounded by αR). This was the reason Chekuri, *et. al.,* modified LOCALCUTS using a uniform bound B at each mother node. The new algorithm, BOUNDEDCUTS, fragments POT based on three parameters α, β, and B that satisfy $\beta \geq \alpha > 1$ and $L^{\star} \leq B \leq (1+\epsilon)L^{\star}$ $\epsilon > 1$. This algorithm cuts off light edges in a manner similar to LOCALCUTS. But it collapses edges based on αB bound. Reader is referred to [3] on how the value of B is chosen .

ALGORITHM BOUNDEDCUTS.
while there exist a mother node m **do**
  Partition children of m into sets N_1, N_2,
   such that child $j \in N_1$ iff $\frac{p_j}{c_{mj}} \geq \beta$
  Cut(m, j) for all $j \in N_1$
  if $R_m + \sum_{j \in N_2} [p_j - c_{mj}] \leq \alpha B$
   then Collapse(m, j) for all $j \in N_2$
   else Cut(m, j) for all $j \in N_2$
end while

We extend BOUNDEDCUTS algorithm choosing the proper value for α and β.

Theorem 2 *The worst case performance ratio of* BOUNDEDCUTS *followed by* LPT *algorithm for scheduling on two uniform processor is* $2.35(1+\epsilon)$.

Proof: From [4] we know, $C \leq \frac{2}{\beta-1}W + \frac{\beta-\alpha}{\alpha-1}C^\star$ for BOUNDEDCUTS algorithm. Since $L = W + C$ we have, $L \leq \max(\frac{\beta+1}{\beta-1}, \frac{\beta-\alpha}{\alpha-1})L^\star$.

We also have $M \leq (1+\epsilon)\alpha L^\star_{max}$ [4]. Using lemma 1 we get, $\frac{T}{T^\star} \leq \max((1+\epsilon)\alpha, 1.5\frac{\beta+1}{\beta-1}, 1.5\frac{\beta-\alpha}{\alpha-1})$. Because $1.5\frac{\beta+1}{\beta-1}$ is strictly decreasing in β, $1.5\frac{\beta-\alpha}{\alpha-1}$ is strictly increasing in β and decreasing in α, and $(1+\epsilon)\alpha$ is strictly increasing in α, in order to minimize the right hand side of the above inequality, the values of these functions must be equal. We thus have $\alpha = 2.35$ and $\beta = 4.51$. Therefore, the performance ratio is $(1+\epsilon)2.35$. $\square$

Similar to [4] we can prove the performance ratios of above two algorithms are tight.

4. P-uniform processor scheduling

In this section we first extend LOCALCUTS heuristic followed by LPT algorithm for a frequent case of the problem. We then prove that the combination of LOCALCUTS followed by any scheduling algorithm with a constant bound on its performance ratio (i.e., LPT or Multifit [8]), provides a performance ratio with a constant bound.

4.1 The frequent case

If W is distributed approximately uniformly among nodes of the POT, maximum degree is considerably less than the number of nodes n, and also we have $p < n$. We can prove that LOCALCUTS followed by LPT has the worst case performance ratio in the interval of $[3 \cdots \frac{3+\sqrt{17}}{2})$. These conditions often occur in scheduling parallel database queries. In such cases, POT is usually a binary or at most 3-ary tree [10] and the costs of pipelined operations do not differ very much. This is because the heavy operators are usually partitioned among some processors (partitioned parallelism). Many of the parallel database applications such as data mining systems often process sophisticated queries whose POTs have high number of nodes. These conditions can be denoted mathematically as $W \geq 2(p-1)R$.

Lemma 2 *Assume a uniform processor system with* p *processor and a* POT $\mathcal{P}$ *in which* $L^\star \geq 2(p-1)R$*. If a fragmentation algorithm fragments* $\mathcal{P}$ *such that* $M \leq k_1 R$ *and* $L \leq k_2 L^\star$*, then applying* LPT *on the produced fragments gives a performance ratio of* $\max(k_1, (2-\frac{1}{p})k_2)$.

Proof: Similar to proof of lemma 1, we have,

$$\forall 1 \leq j \leq p \,:\, T \leq \frac{L^{'}_j + M_k}{s_j}$$

which leads to

$$T \leq \frac{L^{'}}{\sum_{j=1}^{p} s_j} + \frac{(p-1)M_k}{\sum_{j=1}^{p} s_j}.$$

Based on the values of k,

1. If $k \geq p$, similar to proof of lemma 1, we can prove that $T \leq k_2 T^\star + \frac{(p-1)k_2}{k}T^\star$. Since $k \geq p$, we have $T \leq k_2 T^\star + \frac{(p-1)k_2}{p}T^\star$. Then, $T \leq k_2(2-\frac{1}{p})T^\star$.

2. If $k < p$, we have $L^{'} = \sum_{l=1}^{k} M_l \leq \sum_{l=1}^{p-1} M_l$. Since M is the heaviest fragment, we have $L^{'} \leq (p-1)M$. Thus, $T \leq \frac{(p-1)M}{\sum_{j=1}^{p} s_j} + \frac{(p-1)M_k}{\sum_{j=1}^{p} s_j}$. And because $M_k \leq M$, we conclude $T \leq 2(p-1)M/\sum_{j=1}^{p} s_j$. From assumption of the lemma, we have $T \leq 2(p-1)k_1 R/\sum_{j=1}^{p} s_j$. And since we have $L^\star \geq 2(p-1)R$, we conclude that $T \leq \frac{k_1 L^\star}{\sum_{j=1}^{p} s_j}$, and thus $T \leq k_1 T^\star$.

Therefore, for the general case we have,

$$T \leq \max(k_1, (2-\frac{1}{p})k_2)T^\star.$$

$\square$

Theorem 3 *The worst case performance ratio of* LOCALCUTS *followed by LPT for scheduling the* POT $\mathcal{P}$*, in which* $W \geq 2(p-1)R$*, on a uniform processor system lies in the interval of* $[3 \cdots \frac{3+\sqrt{17}}{2})$*, depending on number of processors.*

Proof: Because of $W \leq L^\star$, we can use lemma 2. Similar to the proof of theorem 1, we can prove that

$$\frac{T}{T^\star} \leq \frac{1}{2}(3 - \frac{1}{p} + \sqrt{17 + \frac{1}{p^2} - \frac{10}{p}}),$$

by selecting

$$\alpha = \frac{1}{2}(3 - \frac{1}{p} + \sqrt{17 + \frac{1}{p^2} - \frac{10}{p}}).$$

This performance ratio always lies in the interval $[3 \cdots \frac{3+\sqrt{17}}{2})$. $\square$

Similar to the previous case, we can show that this performance ratio is tight.

4.2 The general case

To solve the problem in the general case, we use a mapping between an arbitrary and the optimal schedule.

Lemma 3 *: Suppose that there exists a mapping function* f *from fragments produced by a fragmentation* π *of the POT* $\mathcal{P}$ *to fragments of the optimal schedule, such that it satisfies the following conditions:*

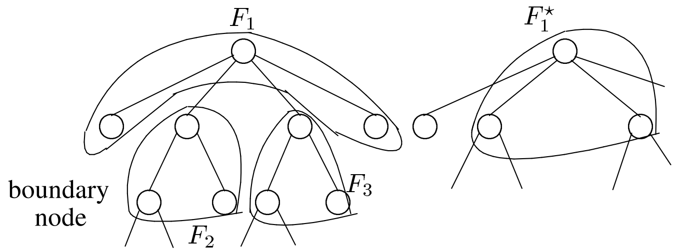


Figure 2. The mapping between LOCALCUTS and optimal fragmentation.

1. *f is total on its domain, and*
2. *for all F_k $k \geq 1$, such that $f(F_k) = F_j^\star$ $j \geq 1$, we have $\frac{\sum M_k}{M_j^\star} \leq r$ ($r \geq 1$).*

Then, scheduling the fragments of π using an uniform processor scheduling algorithm, A_s, with the worst case performance ratio of ϵ has performance ratio less than or equal to $r\epsilon$.

Proof: First, we make a schedule S_1 from the optimal schedule of π. We assign each F_k fragment of π to the processor that $f(F_k)$ has been assigned to in the optimal schedule. Because of totality of f, we can schedule all fragments of π in this manner. From our assumption, if T_1 is the response time of S_1, we have $\frac{T_1}{T^\star} \leq r$. Clearly, if $T_1^\star$ is the response time of the optimal schedule of fragments produced by π, we have $T_1^\star \leq T_1$. Therefore, scheduling these fragments by A_s yields a response time of $T \leq \epsilon T_1^\star \leq \epsilon T_1$. Therefore, $\frac{T}{T^\star} \leq r\epsilon$. □

Definition 1 *The main node of a fragment F is a node whose parent does not belong to F. The main node of F is denoted by $\mu(F)$.*

In other words, the main node of a fragment is the highest level node in that fragment. Clearly, every fragment has one and only one main node.

Theorem 4 *The worst case performance ratio of* LOCALCUTS *followed by the uniform processor scheduling algorithm that has the worst case performance ratio of ϵ is 8ϵ.*

Proof: We define relation f from fragments of LOCALCUTS to those of optimal solution such that $f(F_k) = F_j^\star$, if and only if $\mu(F_k)$ belongs to F_j. Clearly, f is a total function on its domain because each fragment of LOCALCUTS has one and only one main node and each node of the POT belongs to one and only one fragment in its optimal solution.

The value of $r = \frac{\sum M_k}{M_j^\star}$, such that $\forall k\ f(M_k) = M_j^\star$, is maximized when $\sum M_k$ is maximized and $M_j^\star$ is minimized. Assume f maps n fragments $F_1, \cdots$, and F_n to one fragment $F_j^\star$. Because of the definition of f, $F_j^\star$ must have at least n nodes. $F_1, \cdots$, and F_n must be connected because they are mapped into a fragment. $M_j^\star$ is also minimized if $F_j^\star$ cuts all incident edges of the main nodes of $F_1, \cdots, F_n$, except those edges that connect the main nodes to each other (see figure 2). This is because the POT is monotone and for every edge e_{kj} we have, $c_{kj} < R_k - c_{kj}$. $M_j^\star$ is minimized if $F_j^\star$ collapses all edges that connect $\mu(F_1), \cdots$, and $\mu(F_n)$ to each other as is shown in the figure. We therefore have,

$$\sum_{k=1}^{n} M_k = \sum_{k=1}^{n} [m_k + \sum_{j=1}^{q} p_j + \sum_{l=1}^{u} c_l],$$

in which m_k is the weight of $\mu(F_k)$, p_j is the weight of the child node j of $\mu(F_k)$ that belongs to F_k, and c_l is weight of the connecting edge between fragments incident to $\mu(F_k)$. For each boundary node j of F_k, we assume that m_j and p_j are the sum of node weights plus the weights of all edges incident to j that are not in F_k.

Because $F_1, \cdots, F_n$ form a subtree, and each connecting edge between F_k and F_j is considered for computing the cost of both F_k and F_j, we have,

$$\sum_{k=1}^{n} M_k = \sum_{k=1}^{n} [m_k + \sum_{j=1}^{q} p_j] + 2 \sum_{l=1}^{n-1} c_l.$$

Since connecting edges are cut by LOCALCUTS, we get,

$$\begin{aligned} \sum_{k=1}^{n} M_k &< \sum_{k=1}^{n} [m_k + \sum_{j=1}^{q} p_j] + \frac{2}{\alpha} \sum_{k=2}^{n} M_k \\ &< \frac{\alpha}{\alpha - 2} \sum_{k=1}^{n} [m_k + \sum_{j=1}^{q} p_j]. \end{aligned}$$

For $M_j^\star$ we have, $M_j^\star = \sum_{k=1}^{n} [m_k + \sum_{j=1}^{q} c_j]$, in which c_j is the weight of the cut edge j incident to $\mu(F_k)$. Note that j cannot be a connecting edges between the main nodes. Since these edges are collapsed by LOCALCUTS, $M_j^\star \geq \sum_{k=1}^{n} [m_k + \frac{1}{\alpha} \sum_{j=1}^{q} p_j]$.

To compute the ratio, we get

$$\frac{\sum_{k=1}^{n} M_k}{M_j^\star} < \frac{\sum_{k=1}^{n} [m_k + \sum_{j=1}^{q} p_j]}{\sum_{k=1}^{n} [m_k + \frac{1}{\alpha} \sum_{j=1}^{q} p_j]} (\frac{\alpha}{\alpha - 2}) < \frac{\alpha^2}{\alpha - 2}.$$

The above ratio is minimized when $\alpha = 4$, and the minimum value of r is 8. This completes the proof. □

For the scheduling phase, we can use MULTIFIT algorithm since its worst case performance ratio is 1.3 [8].

Corollary 1 *The worst case performance ratio of* LOCALCUTS *followed by* MULTIFIT *algorithm for scheduling on a uniform processor system is* 10.4.

Corollary 2 *The worst case performance ratio of* LOCALCUTS *followed by the PTAS of the uniform processors scheduling of Hochbaum and Shmoys is* $8(1+\epsilon)$.

The first algorithm is more practical compared to the second one. Because MULTIFIT takes $O(n \log n)$, but PTAS has a high running time of $O(n^{\frac{10}{\epsilon^2}+3})$.

5. Conclusions

In this paper we studied the heterogeneity of resources in parallel query scheduling. This is an important subject and has not been considered in previous similar works. We introduced the POT scheduling on the uniform processor system and proposed approximation algorithms for some special cases of the problem as well as for the general case. We showed that our algorithms have constant bounds on the worst case performance ratio. We have also shown by experimentation that the average case performance ratios of our algorithms are near optimal [16].

References

[1] *Bamford, R., Butler, D., Klots, B., and Macnaughton, N. "Architecture of Oracle Parallel Server,"* In Proceedings of 24th Int'l Conference on VLDB, New York, USA, August 1998.

[2] *Brunie, L., Exbrayat, M., Flory, A. "A Coupled Approach for Parallel Evaluation of Relational Queries on a Network of Workstations,"* XIV. International Symposium on Computer and Information Sciences, October 1999, Kusadas, TURKEY.

[3] *Chekuri, C., "Approximation Algorithms for Scheduling Problems,"* PhD. Thesis, Stanford University, Department of Computer Science, 1998.

[4] *Chekuri, C., Hasan, W., and Motwani, R., "Scheduling Problems in Parallel Query Optimization,"* In Proceedings of Fourteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principle of Database Systems, pp: 255-265, San Joes, California,May 1995.

[5] *Dewitt, D.J., and Gray, J, "Parallel Database Systems: The Future of High Performance Database Systems,"* Communication of ACM, 35(6), pp:85-98, June 1992.

[6] *Englert, S., Glasstone, R., and Hasan, W., "Parallelism and Its Price: A Case Study of NonStop SQL/MP,"* Sigmod Record, December 1995.

[7] *Florescu, D., Hasan, W., and Valdurize, P., "Open Issues in Parallel Query Optimization,"* Sigmod Record, 25(3), pp:28-33, September 1996.

[8] *Friesen, D., and Langston, M., "Bounds for* MULTIFIT *Scheduling on Uniform Processors,"* SIAM Journal on Computing, 12(1), pp: 60-70, 1983.

[9] *Gonzalez, T., Ibarra, O., and Sahni, S., "Bounds for LPT Schedule on Uniform Processors,"* SIAM Journal on Computing, 6(1), pp: 150-164, March 1977.

[10] *Hasan, W., "Optimization of SQL Queries for Parallel Machines,"* PhD. Thesis, Stanford University, Department of Computer Science, December 1996.

[11] *Hasan, W., and Motwani, R., "Optimization Algorithms for Exploiting the Parallelism, Communication Tradeoff in Pipelined Parallelism,"* In Proceedings of 20th International Conference on VLDB, pp:36-47, Santigo, Chile, September 1994.

[12] *Hochbuam, D., and Shmoys, D., "A Polynomial Approximation Scheme for Scheduling on Uniform Processors: Using The Dual Approximation Approach,"* SIAM Journal on Computing, 17(3), pp:539-553, June 1988.

[13] *Hong, W., "Parallel Query Processing Using Shared Memory Multiprocessors and Disk Arrays,"* PhD. Thesis, University of California, Berkeley, Department of Computer Science, August 1992.

[14] *Kitsuregawa, M., Oguchi, M., and Tamura, T., "Parallel Database Processing on a 100 Node PC Cluster: Cases for Decision Support Query Processing and Data Mining,"* In Proceedings of International Conference for High Performance Computing and Communications, San Jose, CA, November 1997.

[15] *Kitsuregawa, M., and Tamura, T., "Dynamic Load Balancing for Parallel Association Rule Mining on Heterogeneous PC Cluster System,"* In Proceedings of 25th International Conference on VLDB, Edinburgh, Scotland, September 1999.

[16] A. Termehchy, and M. Ghodsi, "Piplined Operator Tree Scheduling for Heterogeneous Environments," submitted.