# PARALLEL QUERY OPTIMIZATION: PIPELINED PARALLELISM SCHEDULING AND GOLDEN NUMBER

Carmen Elena ODUBĂŞTEANU[1], Călin Aurel MUNTEANU[2]

*Problema planificării paralelismului de tip pipeline este foarte importantă pentru optimizarea interogărilor paralele. Pentru a o rezolva se utilizează ca reprezentare un arbore de operatori pipeline (Pipelined Operator Tree- POT), un arbore ale cărui noduri reprezintă operatorii interogării care pot fi executați în paralel iar muchiile reprezintă costul de comunicare dintre doi operatori adiacenți; trebuie să determinăm un plan de execuție pentru POT care minimizează timpul total de răspuns, aceasta fiind o problemă de tip NP-complexă. Lucrarea de față prezintă algoritmi pentru determinarea planificării paralele de tip pipeline şi compară performanțele acestora din mai multe puncte de vedere prin simularea comportamentului lor. O parte din algoritmi sunt propuşi de autori, doi dintre algoritmi fiind bazați pe utilizarea Numărului de Aur.*

*Pipelined parallelism scheduling problem is very important in the area of parallel query optimization. To model the problem it is used a POT (Pipelined Operator Tree), which is a tree whose nodes represent query operators that can be run in parallel and edges represent communication between adjacent operators; we must find a schedule for the POT that minimizes the total response time, a problem which has been shown to be NP-hard. This paper presents algorithms for pipelined parallelism scheduling and compares their performances by simulating their behaviors. Some of the algorithms are proposed by the authors; two of them are based on Golden Number.*

**Keywords**: query optimization, parallel databases, pipeline parallelism scheduling, Golden Number

## 1. Introduction

Today we are challenged with sophisticated applications on parallel database systems, such as decision support systems and data mining. Therefore, the minimization of the query response time is more than ever necessary. The complexity of this problem is reduced if we used a two-phase approach [1], [2]: join ordering and query rewriting followed by parallelization and scheduling. In the second phase, atomic units of the query (operators) are extracted and then

[1] Assistant, Department of Computer Science, University POLITEHNICA of Bucharest, Romania, e-mail: carmen_od@yahoo.com
[2] Assistant, Department of Automatics and Industrial Information, University POLITEHNICA of Bucharest, Romania, e-mail: mc_aurel@yahoo.com

scheduled to provide the minimum response time. One of the most important issues that must be considered is the parallelism-communication trade-off [3], [4]. A query will be represented as a weighted operator tree in which each node represents an operator and each edge represents the timing constraints between operators [5], [6]. A timing constraint is either a precedence or parallel constraint. The parallel constraint introduces a pipelined parallelism and requires that the two adjacent nodes start and terminate their works approximately at the same time, behaving as a producer-consumer system.

Algorithms for managing pipelined parallelism are an essential component of an optimizer because pipelining is sometimes the only way of speeding up a query not just a useful supplement to partitioned parallelism [3]. For example, when each reduced relation of a query that join a large number (say 10) of relations and apply external functions, grouping and aggregation is small, partitioned parallelism ceases to be a viable option and pipelined parallelism is the only source of speedup.

Scheduling of a Pipelined Operator Tree (POT - weighted operator tree in which all edges represent parallel constraints [6]) is different from the classical scheduling problems because of the communication.

Brute force algorithms are impractical for scheduling pipelines due to the extremely large search space. A query that joins 10 relations leads to an operator tree with about 20 nodes. The number of ways of scheduling 20 operators on 20 processors exceeds $5 \times 10^{13}$. Algorithms that simply ignore communication overhead are unlikely to yield good results. Communication cost is saved if adjacent nodes are assigned to one processor but this would decrease the degree of parallelism.

This optimization problem can also be viewed as to find a schedule that minimizes the maximum load of the processors where load of a processor 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.

POT scheduling problem was first introduced by Hasan and Motwani for identical processor systems and was shown to be NP-hard [1], [6]. They proposed several approximation algorithms. Five of them are presented and compared from different points of view. These algorithms are: Modified LPT, BalancedCuts, Hybrid, LocalCuts and BoundedCuts. Also, in this paper are described four recent algorithms OptimBalancedCuts, OptimHybrid, FiLocalCuts and FiBoundedCuts designed by us. FiLocalCuts and FiBoundedCuts are based on Golden Number.

The paper is organized as follows: it begins with an overview of the model and problem definition. Then, nine algorithms for scheduling pipelines parallelism are presented. Finally, the experimental results are presented and analyzed. Also, a short presentation of Golden Number is made in this paper.

## 2. A model for the problem

The following definitions are based on earlier models presented in [1], [6], [7]. A POT is represented as a weighted operator tree P= (V, E) with n nodes. The weight $t_i$ of the node i is the time to run the operator in isolation assuming all communications are local. The weight $c_{ij}$ of the edge from node i to node j is the additional CPU overhead that both i and j will incur for inter-operation communication if they are scheduled on different processors. A schedule of P on p processors is a partition of V, the set of n nodes, into p sets $F_1, F_2,\ldots, F_p$ 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, $L_k=\sum_{j\in Fk}[t_j+\sum_{l \notin Fk} c_{jl}]$. L is $\max_{1<=k<=p} L_k$.

Two operations are used to modify the POT: collapse (i,j) is to replace adjacent nodes i and j by a single node i` having weight of $t_{i`}=t_i+t_j$. Operation cut (i,j) is to delete edge (i,j) and add its weight to those of node i and j. Collapse and cut operations should be interpreted as decisions to allocate nodes to the same or distinct processors respectively.

As shown in [6], we can convert each POT into a POT with no worthless edges, called monotone tree, by collapsing all its worthless edges using the GreedyChase algorithm that "chases down" and removes parallelism that is "worthless" irrespective of the number of processors. A GreedyChase algorithm is used as a pre-processing step in all described algorithms. Then we schedule the monotone tree. In a monotone tree we use also the following notations: $R_i=t_i+\sum_{j\in V} c_{ij}$, $R=\max_{1<=i<=p} R_i$ and $W=\sum_{i\in V} t_i$.

## 3. Pipelined scheduling algorithms

Scheduling pipelined operator tree is an intractable problem [8] and the space of schedules is super exponentially large. Thus any algorithm that finds the optimal is likely to be too expensive to be usable. The following algorithms are fast heuristics that produce near-optimal schedules.

### Modified LPT Algorithm

Modified LPT algorithm [9] simply preprocesses away worthless parallelism by running GreedyChase before running LPT [8]. LPT assigns the job with the largest running time to the least loaded processor, repeating this step until all jobs are assigned.

The algorithm is still oblivious to the tradeoff between parallelism and communication. Edges in a monotone path can have high weights and the algorithm is unaware of the savings that can occur when two nodes connected by an edge with a large weight are assigned the same processor.

**BalancedCuts algorithm**

BalancedCuts algorithm [9], is finding the optimal connected schedule. A connected schedule requires the nodes assigned to any processor to be a connected set.

The algorithm for finding the optimal connected schedule for trees in which all edge weights has two steps repeated until the resulting number of fragments is no more than p:

1. B will be set to a lower bound on the response time

2. Given a bound B and a number of processors p, the BPSchedule algorithm [8] will be run to find a connected schedule with a response time of at most B, if such a schedule exists. Algorithm simply picks a mother node (a node is a mother node if all adjacent nodes with at most one exception are leaves) and traverses the children in the order of non-increasing $t_i$ - $c_{im}$. Then, children are collapsed into the mother node as long as the weight of the mother stays below B and then cut off the rest. The process is repeated until no more mother nodes are left. If the resulting number of fragments is no more than p, a (B,p)-bounded schedule was found, otherwise no such schedule is possible.

For an unsuccessful run of BpSchedule we will revise B as being the minimum of $B_i$ (for each fragment $F_i$ produced by BpSchedule, let $B_i$ be the cost of the fragment plus the weight of the next node that was not included in the fragment).

**OptimBalancedCuts algorithm**

OptimBalancedCuts algorithm is an optimization of the BalancedCuts algorithm introduced by authors in [10]. A more careful analysis (and implementation) of the following idea gives us a bound of O(np). Whenever the B value is updated, the total work done in finding a new candidate solution can be charged to the nodes which migrate from a component to a previous one. It is easy to verify that the implementation cost works out to be O(1) for each such node migration. Since any one node can migrate at most p times, the total work can be bounded by O(np).

The algorithm picks a mother node, traverses the children in the order of non-increasing $t_i$ - $c_{im}$, collapses them into the mother node as long as the weight of the mother stays below B and then cut off the rest saving for each node the context (defined by the current mother node, the current son, the tree and the number of cuts) before the corresponding cutting step. The process is repeated until no more mother nodes are left or the number of cuts is not p-1.

If the cost of the last fragment is no more than B, a (B,p)-bounded schedule was found, otherwise no such schedule is possible.

B is revised by minimum of $B_i$ and we repeat the process of collapsing nodes to their mothers beginning from the context corresponding to the node C

(the one choused for the new B value, which has the minimum value from the last iteration cutting nodes). So, the algorithm is run directly from the iteration corresponding to the mother node of C, skipping that way the steps already made before the cutting of the node C.

**Hybrid Algorithm**

BalancedCuts performs poorly on stars since the constraint of connected schedules is at odds with load balancing [9]. While the algorithm is cognizant of communication costs, it is poor at achieving balanced loads. On the other hand, LPT is very good at balancing loads but unaware of communication costs. The Hybrid algorithm [9] resulted by combining these two algorithms: BalancedCuts to cut the tree into many fragments and then schedule the fragments using LPT. LPT can be expected to "cleanup" cases such as stars on which connected schedules are a bad approximation.

**OptimHybrid Algorithm**

Hybrid algorithm has the best performance ratio in our experiments. Also, it has the worst execution time. So, we developed OptimHybrid algorithm [11], based on Hybrid, which has a better complexity. OptimHybrid uses OptimBalancedCuts algorithm instead of BalancedCuts algorithm to cut the tree into fragments which are then scheduled using LPT.

Algorithm OptimHybrid:

1. T' = GreedyChase
2. for i = p to n do
3. $F_1,\ldots , F_i$ = OptimBalancedCuts(T', i)
4. schedule = LPT($\{F_1,\ldots , F_i\}$,P)
5. end for
6. return best of schedules found in steps 2 to 5

**Approximation algorithms**

For approximation algorithms we use a two-stage approach: fragmentation, and the actual scheduling. For scheduling, it was used LPT algorithm.

In order to obtain better performances we used the Golden Number, (known also as Fibonacci number, Divine section, Phi or Φ) which has an approximate value of 1.618. This number is often met all around the world, from the ancient and modern art and architecture to the organization of nature, including human beings too. It is said that Phi represents a measure for harmony, a divine proportion known and used from the antiquity. In the next section a short presentation of Golden Number is introduced.

*LocalCuts and FiLocalCuts algorithms*

LocalCuts [1] 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 $\alpha$, 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 FiLocalCuts algorithm we used for $\alpha$ Golden Number based values.

Algorithm FiLocalCuts:

1. T' = GreedyChase
2. while there exists a mother node m with child j do
3. If $t_j > \alpha c_{jm}$ then cut(j,m)
4. else collapse(j,m)
5. end-while
6. return schedule

*BoundedCuts and FiBoundedCuts algorithms*

The second algorithm that we modify is BoundedCuts [1]. If R is small compared to M*, LocalCuts may cut expensive edges needlessly (maximum weight of fragments produced by LocalCuts is bounded by $\alpha R$). It uses a uniform bound B for each mother node. BoundedCuts fragments POT based on three parameters $\alpha$, $\beta$ and B that satisfy $\beta \geq \alpha > 1$ and $B \geq R$. This algorithm cuts off light edges in a manner similar to LocalCuts. But it collapses edges based on $\alpha B$ bound.

We extended BoundedCuts algorithm choosing in FiBoundedCuts values based on Golden Number for $\alpha$ and $\beta$.

Algorithm FiBoundedCuts:

1. while there exist a mother node m do
2. partition children of m into sets $N_1$ and $N_2$ such that child $j \in N_1$ iff $t_j / c_{mj} \geq \beta$
3. cut(m,j) for all $j \in N_1$ ; ($\beta$ rule)
4. if $R_m + \sum_{j \in N2}( t_j - c_{mj} ) \leq \alpha B$ then collapse(m,j) for all $j \in N_2$ ;
5. else cut(m,j) for all $j \in N_2$ ; ($\alpha$ rule)
6. end-while
7. return schedule

## 4. Experimental Results

Based on experimental data, the presented algorithms will be compared in this section from different points of view.

Over 5000 trees were generated for experiments, from which, after applying the algorithm for conversion into monotone trees, only those who have 10 nodes were kept, meaning 1000 trees. The generated trees have a value domain ranging from 1 to 20 both for each node weight and for edge's weight. All presented algorithms were simulated for every monotone tree and for a number of processors p ranging from 1 to maximum 10. The approximation algorithms were tested for different values of their parameters, including Golden Number based values.

For performance analysis of each algorithm, performance ratio was defined as the ratio between experimental value and optimal value. The optimal value was considered the maximum values from $R=\max_{i\in V}R_i$ and $(W+C_E)/p$ where $C_E$ is the sum of the weights of the cheapest p-1 edges [9].

### Performance ratio

The results for average performance ratio are presented in a graphical (comparison) form in Fig. 1.

The most efficient algorithms are Hybrid and OptimHybrid (same performance ratio). Also, Modified LPT, FiBoundedCuts and FiLocalCuts present good performances, closed to Hybrid.

In Fig. 2 we have a graphical comparison for LocalCuts and FiLocalCuts algorithms tested with different values for parameter α.

Notice that:

- Best values (the smallest ones) for performance ratio are obtained for Φ/2, for p≥4.
- For α=Φ, medium performance ratio is relatively very good for any number of processors (performance ratio is in domain [1, 1.18]; for Φ/2 domain is [1, 1.22]).
- Same minimum interval [1, 1.18] is obtained for α in [1.6, 2] but, more closed parameter value is by Φ better performance ratio are obtained for p ≥3.
- For p=2 we have good performances for bigger values of parameter α (for example, for α=3.56 performance ratio is 1.1; for Φ is 1.17).

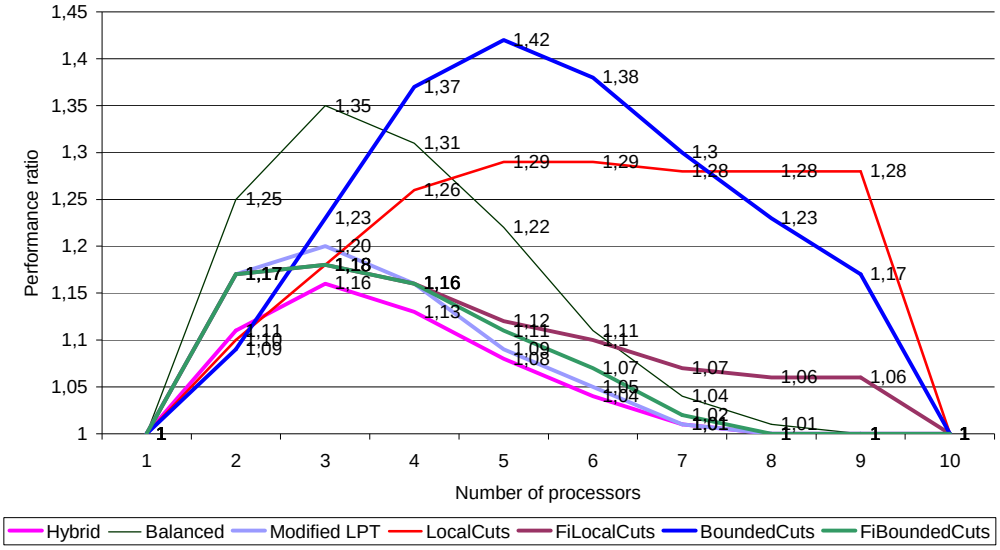


Fig. 1. Performance Ratio for Modified. LPT, BalancedCuts, Hybrid, LocalCuts, FiLocalCuts, BoundedCuts, FiBoundedCuts

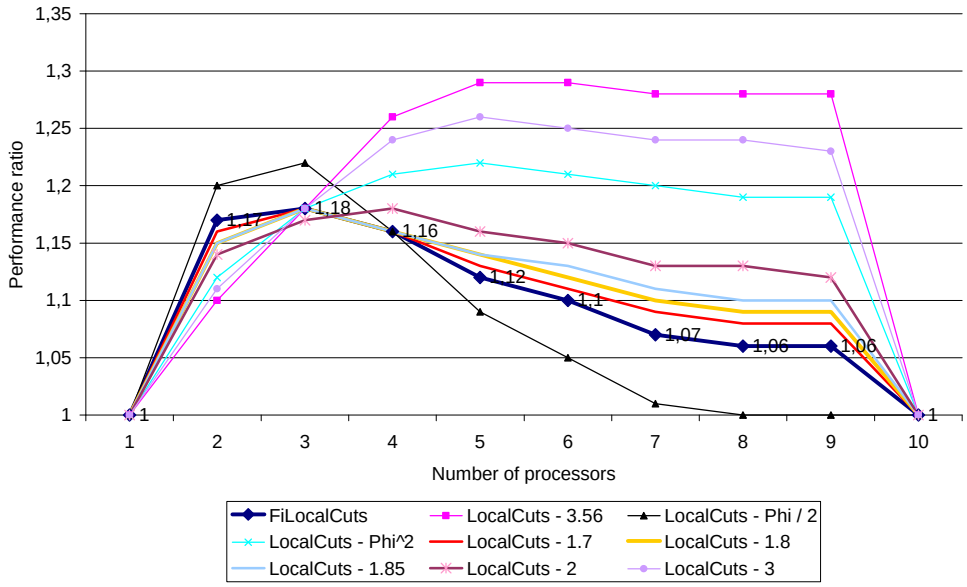


Fig. 2. Performance Ratio for different values of parameter α for LocalCuts, FiLocalCuts

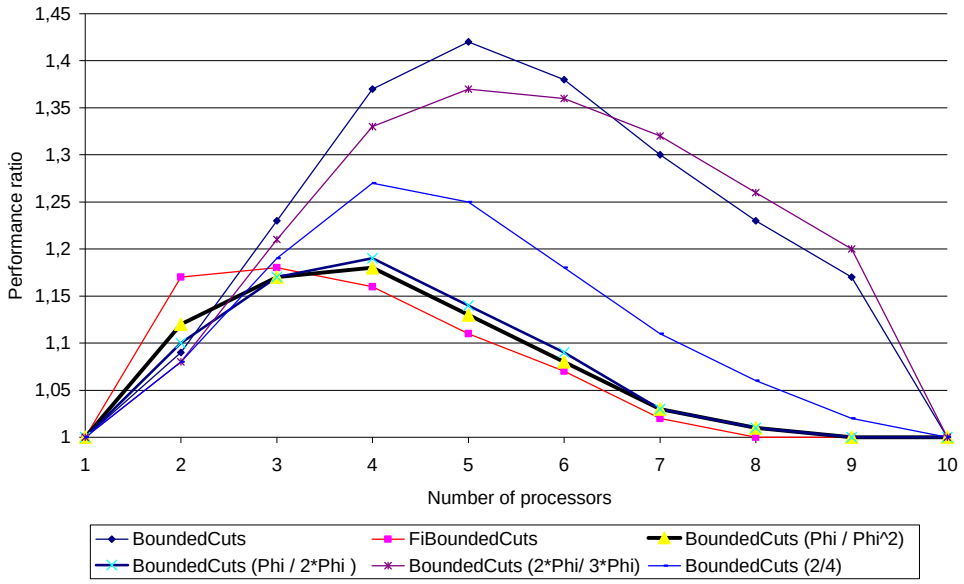


Fig. 3. Performance Ratio for different values of parameters α, β for BoundedCuts, FiBoundedCuts

Values for performance ratio for BoundedCuts and FiBoundedCuts for different values of parameters α, respectively β are presented in Fig. 3.

Notice that:

- Best values (the smallest ones) for performance ratio are obtained for $\alpha=\Phi$ and $\beta=\Phi$, for $p\geq 4$.
- For $p = 3$ best values are obtained for $\alpha=\Phi$ and $\beta=2\Phi$.
- For $p=2$ very good values are obtained for $\alpha=\Phi$ and $\beta=3\Phi$.
- For bigger values of β we have better performance ratio for a small number of processors and for smaller values of β for a bigger number of processors ($p \geq 4$).

Also, the performance ratio for original algorithms LocalCuts, BoundedCuts and newly FiLocalCuts and FiBoundedCuts are presented in Fig. 4.

In conclusion, FiLocalCuts and FiBoundedCuts present better performance ratio than LocalCuts, respectively BoundedCuts, the algorithms which are derived from and the use of Golden Number based values for parameters α and β was a good idea.

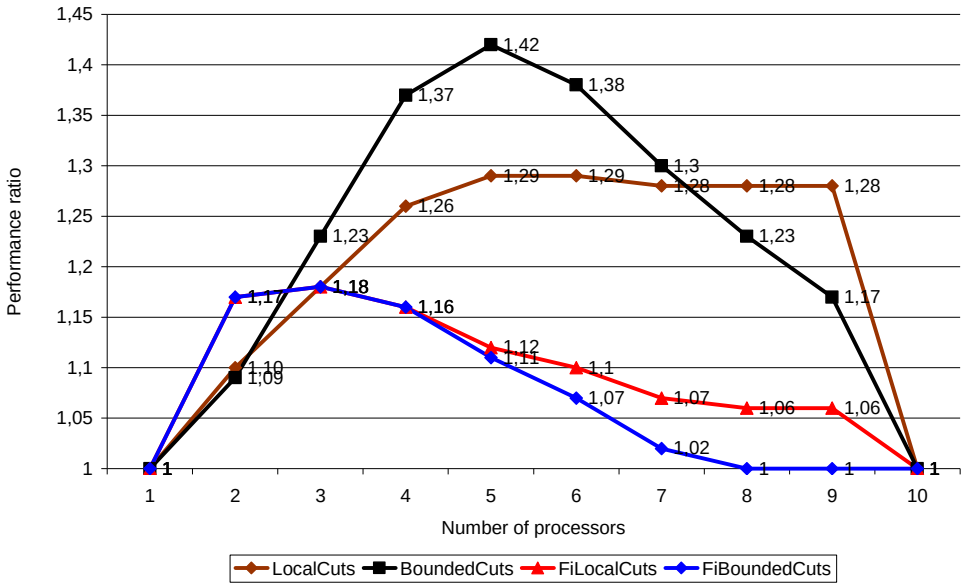


Fig. 4. Performance Ratio for LocalCuts, FiLocalCuts, BoundedCuts, FiBoundedCuts

**Minimum, respectively maximum values and domain range for performance ratio**

For all the 1000 tested trees, the minimum performance ratio value was 1 for all the presented algorithms. The maximum performance ratio values (and the number of processors for which are obtained) are in table 1. Also, the interval length for performance ratio is detailed.

Hybrid and OptimHybrid present the minimum for both maximum performance ratio (1.53) and domain length (0.53). Same values are obtained by FiBoundedCuts which dramatically improved BoundedCuts performances (worst values both for performance ratio (2.42) and domain length (1.42)).

A closed value (0.59) for domain length is also obtained by FiLocalCuts which presents a significant improvement from this point of view relative to LocalCuts (1.15).

*Table 1*

**Performance Ratio (max, max-min)**

| Algorithm | P | Max | Max-Min |
|---|---|---|---|
| ModLPT | p = 3 | 1.74 | 0.74 |
| BalancedCuts, OptimBalanced | p = 3 | 1.84 | 0.84 |
| Hybrid, OptimHybrid | p = 3 | 1.53 | 0.53 |
| LocalCuts | p = 4, 5, 6, 9, 10 | 2.15 | 1.15 |
| BoundedCuts | p = 4, 5 | 2.42 | 1.42 |
| FiLocalCuts | p = 7, 8 | 1.59 | 0.59 |
| FiBoundedCuts | p = 4 | 1.53 | 0.53 |

### Average time for generating the solution

From this point of view, the following values (presented in the next figure and table) were obtained:

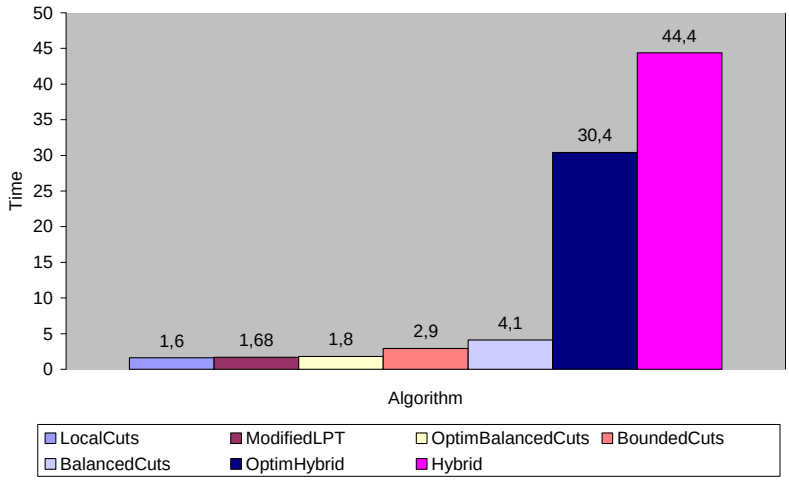


Fig. 5. Execution Time

The best time is obtained by LocalCuts and FiLocalCuts.

Notice that OptimBalancedCuts has an execution time (1.8) much better than BalancedCuts (4.1), the algorithm which is derived from. So, the execution time for OptimHybrid was also decreased with approximately 32% reporting to the Hybrid execution time.

*Table 2*

**Execution Time**

| Algorithm | Execution time | | Algorithm | Execution time |
|---|---|---|---|---|
| LocalCuts, FiLocalCuts | 1.6 | | BalancedCuts | 4.1 |
| Modified LPT | 1.68 | | OptimHybrid | 30.4 |
| OptimBalancedCuts | 1.8 | | Hybrid | 44.4 |
| Boundedcuts, FiBoundedCuts | 2.9 | | | |

### Algorithm complexity

From this point of view, the values presented in table 3 were obtained.

*Table 3*

**Algorithms Complexity**

| Algorithm | Complexity | | Algorithm | Complexity |
|---|---|---|---|---|
| Modified LPT, UniformModLPT | O(nlog(n)) | | BalancedCuts | $O(n^2p)$ |
| LocalCuts, FiLocalCuts | O(nplog(n)) | | OptimHybrid, UniformHybrid | $O(n^2p)$ |
| BoundedCuts, FiBoundedCuts | O(nplog(n)) | | Hybrid | $O(n^3p)$ |
| OptimBalancedCuts | O(np) | | | |

Notice that OptimBalancedCuts complexity (O(np)) was decreased reporting to the BalancedCuts complexity ($O(n^2p)$). Also, OptimHybrid and UniformHybrid complexity ($O(n^2p)$) was decreased reporting to the Hybrid complexity ($O(n^3p)$), due to the decreasing of OptimBalancedCuts complexity.

## 5. Conclusions

Nine algorithms regarding inter-operator parallelism (processors allocation phase for pipeline operator trees) are presented and analyzed in this paper. From [9], Modified LPT, BalancedCuts and Hybrid; from [1], LocalCuts and BoundedCuts, then OptimBalancedCuts and OptimHybrid designed by authors in [10], [11] and two new extended algorithms FiLocalCuts and FiBoundedCuts.

From the simulations results, the best performance ratio is obtained by OptimHybrid very closed by FiLocalCuts, Modified LPT and FiBoundedCuts performances.

## Annexes A – Golden Number

The Golden Number [12], or Golden Ratio, or Golden Mean or Fibonacci Number is one of these mysterious irrational numbers, like *e* or π(Pi). It is often called Φ or Phi. Φ is said to be the divine proportion, the ratio of beauty. Indeed, it has been found in some nature constructions, later re-used in architecture and paintings.

### The value of Φ

The positive result of the equation "$X^2=X+1$" gives Φ value:

$\Phi = (1+\sqrt{5})/2 =$ **1,618**0339887498948482 0...

$\varphi = (1-\sqrt{5})/2 = -0,6180339887498948482 0...$

Φ is also the limit of the ratio of Fibonacci series numbers:

$$\lim_{n \to \infty} \left( U[n+1] / U[n] \right) = \Phi,$$

Where U **(**Fibonacci series) is defined as follow:

U[n+2]=U[n+1]+U[n], U[0] = 0, U[n]=1.

U[n] values are: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, etc.

### Mathematical properties of Φ

In both arithmetic and geometry, Φ has many properties that we cannot develop here. The basic knowledge to keep in mind with the Golden Number is:

$\Phi^2 = \Phi + 1$.

Φ and φ are very close as far as their decimal part is identical and they have the following properties:

$\Phi * \varphi = -1$,

$\Phi + \varphi = 1$, etc.

Geometrically, Φ is defined as in Fig. 6.

Euclid (~300BC) called this geometrical drawing "to divide a line in mean & extreme ratio".

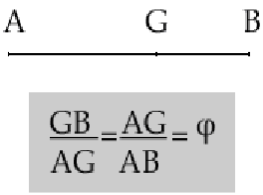


Fig. 6. Golden Section

It is important to know that Φ is particularly present in the geometry of Pentagon (2D), Pentagram (2D) and Dodecahedron (3D) (Fig. 7):

Fig. 7. Pentagon, Pentagram, Dodecahedron

The use of Golden Number, Phi for FiLocalCuts and FiBoundedCuts improves LocalCuts, respectively BoundedCuts performances for both maximum performance ratio and its domain length. Thus, this first use of Golden Number in pipelined parallelism scheduling problem was a successful one and it proves that Golden Number implications in pipelined parallelism are alike to the implications which appears in other domains (an “harmonious distribution” for performance ratio values was obtained).

**The Golden Number everywhere**

Here are some famous examples of use of Φ in the nature:

- The Nautilus shell (Nautilus pompilius) grows larger on each spiral by Φ (Fig. 8)
- The sunflower has 55 clockwise spirals overlaid on either 34 or 89 counterclockwise spirals, a Φ proportion
- For a coneflower (Fig. 9) you can see that the orange "petals" seem to form spirals curving both to the left and to the right. At the edge of the picture, if you count those spiraling to the rights as you go outwards, there are 55 spirals. A little further towards the centre and you can count 34 spirals. You will see that the

pair numbers (counting spirals in curing left and curving right) are neighbours in the Fibonacci series.

- The drone genealogy follow the Fibonacci series (so is linked to Φ), etc.

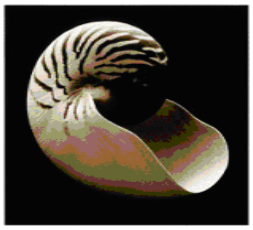

Fig 8. Nautilus shell

Fig 9. Coneflower - a member of the daisy family with the scientific name Echinacea purpura

Φ has also been used by artists (painters, sculptors...) who were trying to rationalize aesthetic and understand what make us think whether a shape is nice / harmonious / pleasant or not. It has been used at Keops in Egypt, at the Parthénon in Athens, at Epidaure (Greece), etc. See Fig. 10, Fig. 11 and Fig. 12.

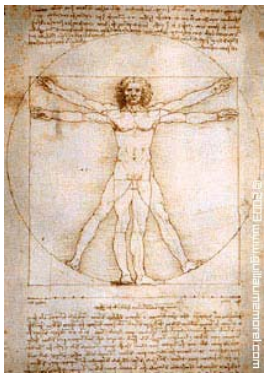

Fig. 10. Leonardo Da Vinci's Corpo Humano, a good illustration of artists in the quest of rationalizing beauty

**How to use quickly the Golden Number?**

When you take the first ratio coming from Fibonacci series, for instance 5/3=1.66..., 8/5=1.6 and 13/8=1.625, you have an approximation of Φ for not precise works (for instance painting). Squaring of 3, 5 or 8 are easy to draw when preparing material for painting, and the previous ratio can help you quickly use "divine proportions".

Another solution to quickly use Golden Number is to use a Golden Compass, as provided by Robert Losson for example.

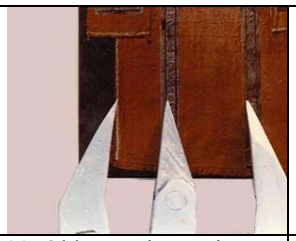

Fig. 11. Old Egyptian tunic

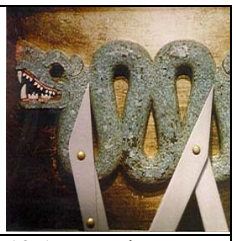

Fig. 12 Aztec culture

Where we also met Golden Number:

- architecture
  - The Parthenon and Greek Architecture
  - Modern Architecture
    - The Eden Project's new Education Building
    - California Polytechnic Engineering Plaza
    - The United Nations Building in New York
- art
  - Leonardo's Art
  - Modern Art
    - Graham Sutherland's Tapestry in Coventry Cathedral
  - in fashioning Furniture
- films
- human body
- poetry
  - Stress, Metre and Sanskrit Poetry
  - Virgil's Aeneid
- music
  - Golden sections in Violin construction
  - Did Mozart use the Golden mean?
  - Phi in Beethoven's Fifth Symphony?
  - Bartók, Debussy, Schubert, Bach and Satie
- miscellaneous, amusing and odd places
  - TV stations in Halifax, Canada
  - Turku Power Station, Finland

## R E F E R E N C E S

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

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

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

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

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

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

[7] *A. Termehchy, M. Ghodsi*, Pipelined operator tree scheduling in heterogeneous environments, Journal of Parallel and Distributed Computing, **Vol. 63**, Number 6, June 2003, pp. 630-637(8)

[8] *R.L. Graham, E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan*, Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey. Annals of Discrete Mathematics, 5:287-326, 1979

[9] *W. Hasan*, Optimization of SQL Queries for Parallel Machines, PhD. Thesis, Stanford University, Dec. 1996

[10] *C. Odubăşteanu, C. Munteanu,* Pipelined operator tree scheduling for heterogeneous and homogeneous environments, ETAI 2007, Ohrid, Macedonia, 2007

[11] *C. Odubăşteanu, C. Munteanu,* Optimization Algorithms for Scheduling Problem in Pipelined Parallelism, Proceedings of the 8th international conference on technical informatics, CONTI, Timişoara, Romania, 2008

[12] www.guillaumemorel.com/en-gold.htm.