International Journal of Information Technology Convergence and Services (IJITCS) Vol.1, No.4, August 2011

# A GENETIC BASED APPROACH FOR MULTI-OBJECTIVE OPTIMIZATION OF DISK SCHEDULING TO REDUCE COMPLETION TIME AND MISSED TASK

R.Muthu Selvi and R.Rajaram

Department of Information Technology

Thiagarajar College of Engineering, Madurai,Tamilnadu,India.

{rmsit,rrajaram}@tce.edu

## ABSTRACT

*Disk scheduling is a method by which a computer operating system decides the order that block I/O operations are submitted to storage volumes. It has an important role in QOS guarantee of soft real-time environments such as video-on- demand and multimedia servers. Many traditional algorithms exist to order the disk requests in an optimized manner. Many of these algorithms attempt to minimize any one parameter, completion-time or missed tasks at a time. In this paper, we propose a genetic based approach for disk scheduling which optimizes the completion time and number of missed tasks simultaneously. Traditional Multi-objective Genetic Algorithm(MOGA) is customized by including the concept of elitism to achieve better fitness values. The existing cycle crossover operator is modified to provide more search space. MOGA hybridized by combining with Simulated Annealing (SA) is known as Hybridized GA(HGA). In this method, the convergence of GA is improved by introducing the probability of SA as the criterion for acceptance of the new trial solutions. This hybridization shows more accuracy while finding solutions in later stage of searching process. The algorithms have been tested on several problems and the results were compared with traditional methods. Experimental results showed that the proposed GA based approaches worked very well and excelled most related works in terms of completion-time and number of missed tasks.*

## KEYWORDS

*Disk scheduling, missed tasks, multi-objective optimization, HGA*

## 1. INTRODUCTION

The use of spinning magnetic disks for data storage has introduced some interesting problems that have greatly challenged computer systems researchers. In a modern computer, the performance of overall system is directly affected by processor speed, memory and disc capacity and disc speed. Although processor speed and memory capacity are increasing by over 40% per year, evolution of disc speed increases more gradually, growing by only 7% per year [1]. Many modern computer applications require huge amounts of data. In some applications data must be retrieved in real-time. Therefore, disk scheduling has great importance in such systems. Examples of such applications may be found in the multi-media field such as video-on-demand and audio playback systems. Disk scheduling is the method by which the computer operating systems decides the order in which block I/O operations will be submitted to storage volumes. The operating system uses a disk scheduling technique to determine which request to satisfy. The I/O scheduler is an operating system component whose purpose is to maximize disk performance and to provide quality of service (QoS) between competing disk users.

Traditionally, disk scheduling problem is tackled in two ways. In one approach, more hardware is added. One example is RAID. It combines multiple physical disks and thereby performance is improved. In another approach, performance and quality of service is improved by improving the software (i.e.) to improve the disk scheduling which efficiently uses the available disk resources.Reordering of disk requests, merging of adjacent requests into single larger requests and delaying a request to the disk drive are the mechanisms used by a disk scheduler.

In this paper, the disk scheduling is formulated as multi-objective optimization problem. MOGA is used to optimize scheduling of the disk requests. The throughput is maximized by minimizing the computational time for the requests and minimizing the number of disk requests missed. This paper is organized as follows: In Section 2 the work related to the problem is discussed. Section 3 describes the problem. Section 4 briefs the methodology of the proposed MOGA. Section 5 presents the experimental results. Section 6 concludes.

## 2. RELATED WORK

Most traditional disk scheduling algorithms, such as FCFS, SCAN, C-SCAN, LOOK, C-LOOK, and SSTF are designed to reduce disk-seek time and increase its throughput [2, 3, 4]. FCFS algorithm performs operations in order the task arrives. However, the performance of this algorithm is poor [5, 6]. SSTF reduces the total seek time compared to FCFS. The disadvantage of SSTF is starvation that is the R/W head stays in one area of the disk if very busy. These algorithms do not consider real-time constraints of I/O tasks and, therefore, are not suitable to be applied directly on a real-time system [6, 7, 8].

In the SCAN algorithm, the disk arm starts at one end of the disk, and moves toward the spindle,servicing requests as it reaches each cylinder, until it gets to the spindle. From this end, the direction of head movement is reversed and servicing continues. The head continuously scans back and forth across the disk [7, 9]. C-SCAN scheduling is a variant of SCAN designed to provide uniform wait time. Like SCAN, C-SCAN moves the head from one end of the disk to the spindle, servicing the requests along the way. When the head reaches the spindle, it immediately returns to the beginning of the disk, without servicing any requests on return trip [9, 10]. LOOK algorithm is also a variant of SCAN. In LOOK the disk head does not move inward or outward when there is no request in that direction. LOOK performs better than SCAN when load is low but it is equivalent to SCAN when the load is high [11]. A variant of LOOK scheduling is C-LOOK. C-LOOK moves the head in one direction from its current position, after serving all the requests in current direction, disk head starts to serve the first request in other end without serving the requests in return trip. It provides more uniform wait time for the requests [11].

Walter A. Burkhard et al [12] considered a novel representation scheme for rotational position optimization reducing the required flash memory by a factor of more than thirty thereby reducing the manufacturing cost per drive. The results indicated the existence of workload domains where the step function rpo tables provide very acceptable performance.Eitan Bachmat [13] considered the problem of estimating the average tour length of the asymmetric TSP arising from the disk scheduling problem with a linear seek function and a probability distribution on the location of I/O requests. Anna Povzner et al [14] have shown that by reserving disk resources in terms of utilization it is possible to create a disk scheduler that supports reservation of nearly 100% of the disk resources, provides arbitrarily hard or soft
guarantees depending upon application needs, and yields efficiency as good as or better than best-effort disk schedulers tuned for performance. Timothy Bisson et al [15] have proposed to use the flash memory to reduce write latency by selectively caching write requests to the NVCache. Hai Huang et al [16] proposed a novel technique that dynamically places copies of data in file system's free blocks according to the disk access patterns observed at runtime.
Teorey et al [17] has performed the analysis on various disk scheduling policies. Thomasian et al [18] has proposed some new disk scheduling policies and analyzed them.Zoran Dimitrijev et al [19] presented Semi-preemptible IO, which divides disk IO requests into small temporal units of disk commands to improve the preemptibility of disk access. The evaluation of this prototype system showed that Semi-preemptible IO substantially improved the preemptibility of disk access with little loss in disk throughput and that pre-emptive disk scheduling could improve the response time for high-priority interactive requests.

Bonyadi [20] proposed a new disk scheduling method based on genetic algorithm that considers make span and number of missed tasks simultaneously. In the proposed method, a new coding scheme is presented that employs crossover, mutation and a penalty function in fitness. Its parameters such as number of chromosomes in initial population, mutation, and crossover probabilities, etc have been adjusted by applying it on some sample problems. The algorithm has been tested on several problems and its results were compared with traditional methods. Experimental results showed that the proposed method worked very well and excelled most related works in terms of miss ratio and average seeks.

## 3. PROBLEM DESCRIPTION

Consider a set of n disk requests r = {r1r2 ... rn}. Finding a feasible schedule r':rw(1) rw(2) rw(y) rw(n) with minimal completion time and minimal number of requests missed is the goal of real- time disk schedulers. The index function w(i), for i=1 to n, is a permutation of {1,2,y, n}. For any disk request, time is required to seek the track where the request is located. This time is called as seektime. It is calculated as given in Equation (1).

$$\text{seektime} = \text{abs}(\text{cheadpos} - \text{tracknum}) \tag{1}$$

where
cheadpos: Current head position
tracknum: Track number of the request
To transfer the request from disk to buffer, some time is spent which is called as transfer time.

The completion time of a request is the time to complete the reuest. It is found using Equation (2).

$$c\_time = cur\_time + seektime + transtime \tag{2}$$

where
c_time : Completion time of the request
cur_time: Current time
seektime: Seek time
transtime: Transfer time
Deadline time is the latest time at which disk request should be completed.
Throughput of the disk scheduling is calculated using Equation (3).

$$\text{Throughput} = b / c\_time \tag{3}$$

where
b: Data size of the request in KB
Since throughput depends on the completion time of the disk requests, our objective is to minimize the completion time. For real-time applications, missed disk requests are to be reduced.

## 4. METHODOLOGY

### 4.1. MOGA

Multi-objective optimization problems often deal with conflicting objectives. Many different approaches have been applied to MOGA problems. Aggregation-based approaches use a weighted sum of the objective values as the new objective in a single-objective optimization problem. Criterion-based approaches consider only one objective of a MOGA problem at a time. In the simplest case, the objectives are ranked in order of importance, optimizing each one in turn without degrading the values of the previous objectives. In this work, the completion time and the number of missed requests are the conflicting objectives. Aggregation-based approach using a weighted sum of the objective values as the new objective in single-optimization problem is used.

### 4.1.1. MOGA Procedure

1. Initialize the population. (Any random ordering of the current requests)
2. Traverse the ordering from left to right
3. Calculate the fitness value
4. Apply Elitism
5.Perform crossover
6.Perform mutation
7. Perform Rank-based selection
8. Repeat steps 3-6 till a required number of generations are reached.

The above steps illustrate how MOGA is implemented. At first, the population is initialized randomly. The fitness value for each individual is calculated according to the fitness function. Elitism is applied to select the individuals to apply the genetic operators. Crossover and mutation are applied on the selected individuals. The individuals are ordered based on the fitness value. The procedure is repeated till convergence occurs.

### 4.1.2. Representation of a disk schedule

A chromosome represents the schedule in which the various disk requests are to be processed. The chromosome is an array of n integers, where n is the total number of disk requests to be scheduled. The allele value at the ith entry of a chromosome represents one disk request in the sequence. The gene represents the disk request id which is an integer. Let 1, 2, 3, 4, 5, 6, 7, 8, 9 and 10 be the disk requests. For this example, the representation of the disk request is shown in Figure 1.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 4 | 2 | 3 | 5 | 1 | 9 | 8 | 10 | 7 | 6 |

Figure 1. Representation of a disk schedule

### 4.1.3. Initialization of the Population

Initially the disk request ids are encoded into integers in a continuous fashion. The disk requests are generated randomly. The population size that we have chosen for experiment is 200.

### 4.1.4. Fitness Calculation

Fitness value is calculated for each of the chromosome in a population as given in (4).

$$\text{Min } f(c\_time, m\_request) = \sum_{i=1}^{n} \alpha * c\_time_i + \beta * m\_request \qquad (4)$$

where

$\alpha$ - Coefficients for completion time, selected as 100.
$\beta$ - Coefficients for number of requests (m_request), 10% of $\alpha$ and is a negative coefficient. Since the completion time affects the throughput and the performance of the disk scheduler, c_time is given more weightage. Hence, $\alpha$ is chosen as 100.

### 4.1.5. Elitism

Elitism is the process of selecting the best chromosomes in a particular generation and retaining them unaffected for the next generation. This is done to overcome the loss of best chromosomes due to the process of crossovers and mutations. The elitism rate that we have chosen is 0.05%.

### 4.1.6. Genetic Operators

Cross over that is used in the experiment is Single-Point cross-over. The cross-over rate used inthe experiment is 90%. Swap mutation with a rate of 1% is used in the work. Based on the fitness value of each chromosome is ranked. As the objective function is to minimize the fitness value, the chromosome with lowest fitness value is assigned the first rank. Chromosome with lowest rank appears in the next generation.

### 4.1.7. Stopping criterion

In most test cases, the convergence is found at 26th or 27th generation. Therefore, the procedure is repeated for 30 generations.

## 4.2. HGA

To develop the hybrid genetic algorithm, we have combined features of SA to the basic MOGA framework. The capability of SA for selecting the fittest candidate solutions are used as input to the cross-over module. Though SA takes some time to cool down to the equilibrium state, it eliminates the dependency of the selection process on a complete pool of candidate solutions required in conventional method at the selection stage. Both SA and MOGA are randomized guided search methods which when combined result in HGA [21].

### 4.2.1. Simulated Annealing

SA is a generic probabilistic metaheuristics for the global optimization problem of locating a good approximation to the global optimum of a given function in a large search space. By analogy with this physical process, each step of the SA algorithm replaces the current solution by a random "nearby" solution, chosen with a probability that depends both on the difference between the corresponding function values and also on a global parameter T, the temperature, that is gradually decreased during the process. The dependency is such that the current solution changes almost randomly when T is large, but increasingly "downhill" (for a minimization problem) as T goes to zero. The allowance for "uphill" moves potentially saves the method from becoming stuck at local optima.
In SA, a problem state is defined by the values of a number of parameters. The state transition is done by changing the values of the parameters using the Boltzmann distribution function in thermodynamics. The objective is to maximize the value of our objective function. At each state transition the temperature of the system is reduced by a small amount. The temperature schedule is designed so that the state of the system freezes after hundreds of transitions. A logarithmically decreasing temperature is found useful for convergence without getting stuck to a local maximum state. But to cool down the system to the equilibrium state it takes time. In particular, simulated annealing knows little about whether a region of the search space has been explored or whether a region is better for searching by the use of statistical distribution function. An interesting and very fast optimization procedure can be developed along with genetic algorithm framework.
For selection, a chromosome with a value xi is taken from a pool P (g) of generation g. It is selected based on Boltzmann probability distribution function. Let it be assumed that fmax is the fitness of the currently available best chromosome. If the next chromosome has fitness f (x) such that it is greater than fmax, then the new chromosome is selected otherwise it is selected with Boltzmann probability as given in Equation (5).

$$\text{Boltzmann probability} = P\left(\exp[-(f(y)-f(x)) / T] >= \text{random } [0, 1]\right) \quad (5)$$

where

$$T = T0\,(1-\alpha)\,k \text{ and } K = (1 + (g / G) * 100) \quad (6)$$

In this equation, 'g' is the current generation number; G, the maximum value of g. The value of α can be chosen from the range [0, 1], and T0 from the range [5,100]. The value of T decreases exponentially or at logarithmic rate with increase in the value of g and hence the value of the probability P. This is significant in terms of convergence. The final state is reached when computation approaches zero value of T, i.e., the global solution is achieved at this point. In the proposed HGA algorithm, the probability that the best string is selected and automatically included as a member of the selected population is very high. However, elitism is suggested to eliminate the chance of any undesired loss of information during the mutation stage. The proposed HGA is shown in Figure 2.

```
Begin
g = 0
initialize (T, P (g))
evaluate P (g) using fitness function
fmax maximum fitness of P (g)
termination_condition = false
while (NOT termination_condition) do
begin
    g = g + 1
    for i = 1 to N do
    begin
        if fmax-f(xj)<=0
        then select xj from P (g) and set fmax to f (xj)
        else if (exp [-(f(y)-f(x)) / T ] >= random [0, 1] )
        then select xj from P (g)
        else select x corresponding to fmax
    end
    crossover
    mutation
    evaluate P (g + 1) using fitness function
    lower T
end
end
```

Figure 2 Proposed HGA

## 5. RESULTS AND DISCUSSION

### 5.1. Simulation Environment and Parameters setting

In the simulation environment, set of read/write requests have been considered that their completion time is calculated using seek time and transfer time. The deadlines of requests are calculated using Equation (7).

$$D = time\_mul \times base + (timeout \times (size / 36KB)) \qquad (7)$$

where

| | |
|---|---|
| D | - Deadline of requests |
| Time_mul | - Time multiplier, uniform random number in interval 1-5. |
| base | - TimeoutBase for each request, 550 |
| timeout | -10 |
| size | -Size of requested data |

The simulation is done in C. Different types of files such as text (.txt), images (.jpg, .bmp), media (.wma, .avi) are given as input. The size of the file is derived and it is used for the calculation of deadline and throughput, it makes suitable for real time environment.

Table 1 Parameters and Values

| **Parameters** | **Values** |
|---|---|
| Number of chromosomes | Number of read/write requests |
| Population size | 200 |
| Elitism Rate | 10% |
| Crossover type | Modified Cycle crossover |
| Crossover Probability | 92% |
| Mutation | Swap mutation |
| Mutation Probability | 1% |

Table 1 gives the various parameters and values used in this experiment. Number of read/write requests decides the number of chromosomes. The population size is fixed as 200. The elitism rate is 10%. Cycle cross over is modified and applied to the selected individuals with the probability 92%. Swap mutation is applied with the probability 1%. With the cross over and mutation types and probabilities as given in Table 1, the experiment is run for various population sizes, 50,100,150,200 and 300. The experiment is run for 1000 times with requests 5, 10 and 15. The results are tabulated in Table 2. At population size 200, the fitness value reaches the lowest level. Though population size is increased to 300, there is no improvement in the lowest fitness value reached. Hence, the population size is fixed as 200 throughout the experiment.

Table 2 Population size vs. Fitness value reached

| Population Size | Proposed MOGA | Proposed HGA |
|---|---|---|
| 50 | 2307 | 2300 |
| 100 | 1802 | 1858 |
| 150 | 1802 | 1800 |
| 200 | 100 | 100 |
| 300 | 100 | 100 |

It is shown that the fitness value reached by proposed MOGA and proposed HGA are almost equal for all the population sizes. For both approaches, the fitness value reaches the lowest level at 200. The comparison is shown in Figure 3.

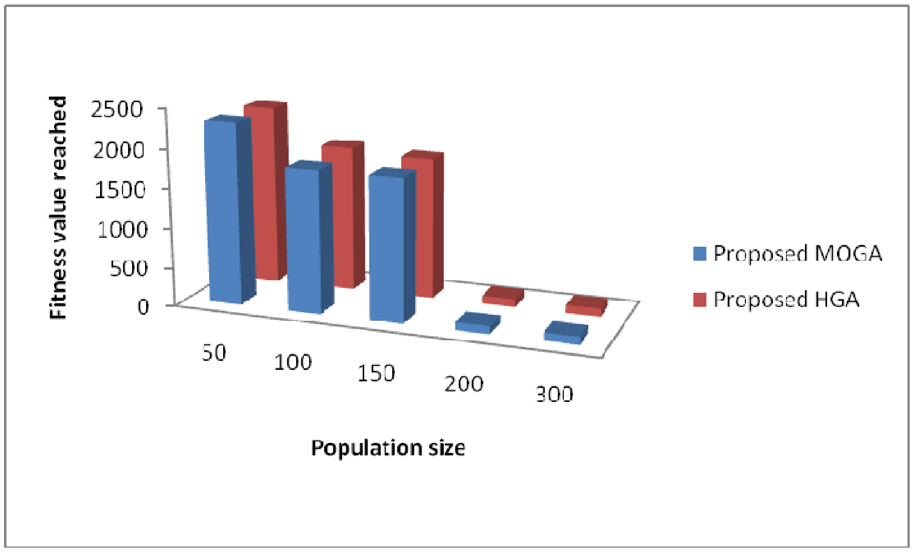

Figure 3 Population size vs. fitness value

## 5.2. Comparison of completion time

The experiment is run for 1000 disk requests. The numbers of disk requests are taken as 5, 10 and 15. The completion times (in ms) of the proposed work obtained through the experiments are tabulated in Table 3.

Table 3 Comparison of completion-time

| **Number of requests** | **FIFO** | **SCAN** | **C-SCAN** | **LOOK** | **C-LOOK** | **SSTF** | **Proposed MOGA** | **Proposed HGA** |
|---|---|---|---|---|---|---|---|---|
| 5 | 500 | 330 | 367 | 317 | 337 | 235 | 250 | 240 |
| 10 | 704 | 365 | 397 | 365 | 352 | 340 | 360 | 360 |
| 15 | 759 | 483 | 520 | 517 | 407 | 415 | 450 | 470 |

Figure 4 illustrates that the completion time is reduced by using the proposed MOGA HGA, when compared to the traditional algorithms, FIFO, SSTF, SCAN, C-SCAN, LOOK, and C-LOOK. The GA based approaches overcome all the existing algorithms except SSTF. The performance of proposed approaches and SSTF in reducing the completion time is equal. However, SSTF does not consider deadline and reduce the missed tasks while scheduling the read/write requests.

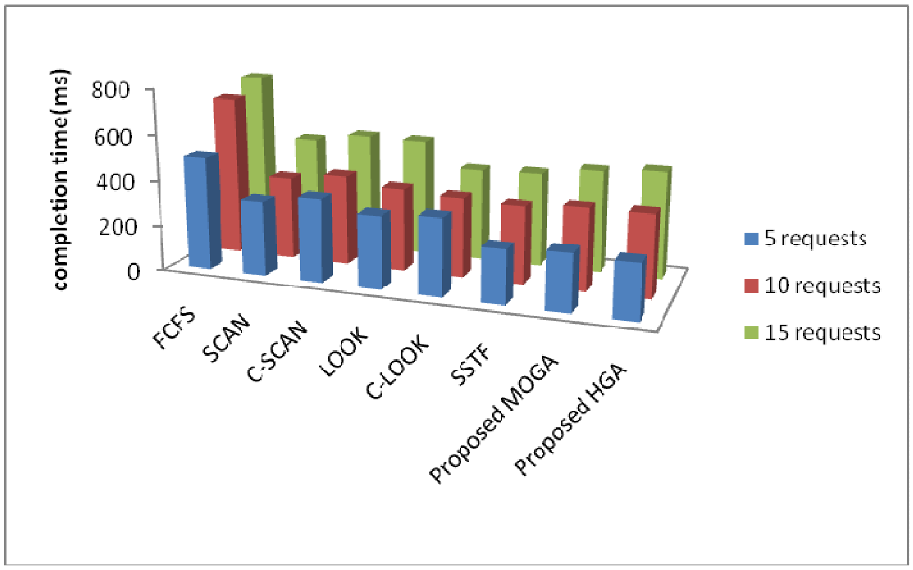

Figure 4 Traditional Algorithms vs. GA based approaches

## 5.3 Pareto Front

Pareto front is the solution set produced by MOGA when the objectivers are controdicting.

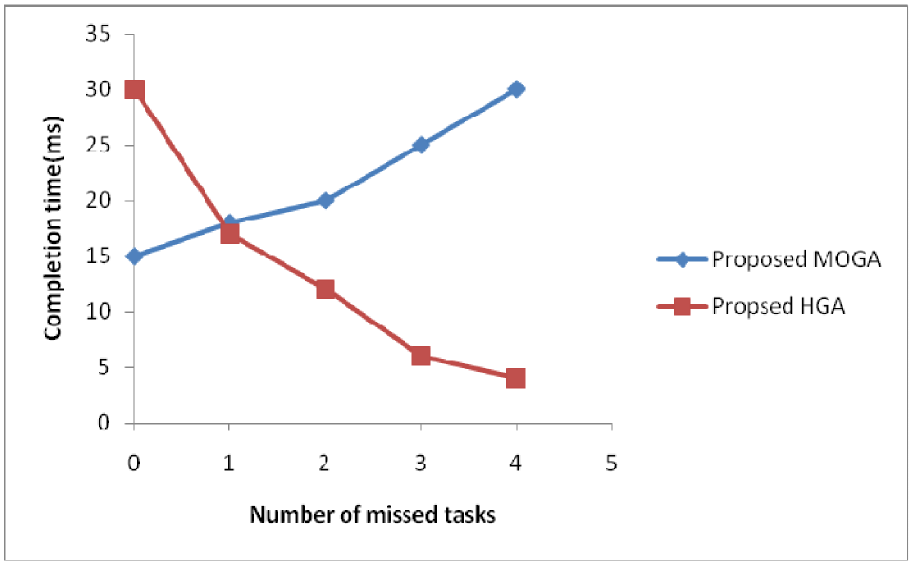

Figure 5 Pareto front

In this work,the number of missed tasks and completion time are the controdictory. It is very crucial to find a trade-off between the two parameters Figure 5 presents the Pareto front produced by the proposed approaches. Through Pareto front, a set of optimal solutions for the disk scheduling problem are found out.

## 6. CONCLUSIONS

In this paper, GA based approaches, MOGA and HGA, are proposed to optimize the disk requests schedule. In the proposed methods, a simple and robust coding technique has been used. To evaluate the proposed method, set of disk requests have been generated randomly. Traditional disk scheduling algorithms are compared with the proposed approaches. The simulation results showed that the proposed approaches produced an optimized schedule versus other related works. It is shown that a trade-off between the conflicting objectives completion time and number of missed tasks is found out by GA based approaches.

## REFERENCES

[1] Lee, S.hyun. & Kim Mi Na, (2008) "This is my paper", ABC *Transactions on ECE*, Vol. 10, No. 5, pp120-122.

[2] Gizem, Aksahya & Ayese, Ozcan (2009) *Coomunications & Networks*, Network Books, ABC Publishers.

[1] C. Ruemmler, J. Wilkes, (1994) "An introduction to disc drive modeling", IEEE Computer Vol.27, No. 3, pp17-29.

[2] Denning, P.J. (1967) "Effects of scheduling on file memory operations", AFIPS Joint Computer Conferences, pp9-21.

3] Geist, R & Daniel, S, (1987) "A continuum of disk scheduling algorithms", ACM Transactions on Computing Systems, Vol.5, No.1, pp77-92.

[4] Chen, S., Stankovic, J.A., Kurose, J.F. and Towsley, D, (1991) "Performance evaluation of two new disk scheduling algorithms for real-time systems", Journal of Real-Time System, Vol.3, No.3, pp307-336.

[5] Seltzer, M., Chen, P. and Ousterhout, J, (1990) "Disk scheduling revisited", USENIX Tech Conference, pp313-324.

[6] Hofri, M, (1980) "Disk scheduling: FCFS vs. SSTF revisited", Communications of the ACM, Vol.23, No. 11, pp645-653.

[7] Chen, T.S., Yang, W.P. and Lee, R.C.T, (1992) "Amortized analysis of some disk scheduling algorithms: SSTF, SCAN and N-Step SCAN", BIT 32, pp546-558.

[8] Worthington, B.L., Ganger, G.R. and Patt, Y.N, (1994) "Scheduling algorithms for modern disk drives", Proceedings of ACM SIGMETRICS Conference, pp241-251.

[9] Coffman, E.G. and Hofri, M, (1982) "On the expected performance of scanning disks", SIAM Journal on Computing, Vol.11, No.1, pp60-70.

[10] Coffman, E.G, (1973) "A note on the relative performance of two disk scanning policies", Information Processing Letters, Vol.2, No. 1, pp15-17.

[11] Sohn, J.M. and Kim, G.Y, (1997) "Earliest-deadline-first scheduling on non-preemptive real-time threads for continuous media server", Proceedings of HPCN, Vol. 1225, pp950-956.

[12] Walter A. Burkhard, John D. Palmer, (2002) "Rotational Position Optimization (RPO) Disk Scheduling", First Conference on File and Storage Technologies.

[13] Eitan Bachmat, (2007) "Average Case Analysis of disk scheduling, increasing subsequences and space-time Geometry", Algorithmica Vol.49, No.3, pp212-231.

[14] Anna Povzner, Tim Kaldewey, Scott Brandt, Richard Golding, Theodore Wong, and Carlos

Maltzahn, (2008) "Efficient Guaranteed Disk Request Scheduling with Fahrrad", Proceedings of the 3rd ACM SIGOPS/EuroSys European Conference on Computer Systems, pp13 - 25.

[15] Timothy Bisson and Scott A. Brandt, (2008) "Reducing Hybrid Disk Write Latency with Flash-Backed I/O Requests", Proceedings of the 2007 15th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, pp402 - 409.

[16] Hai Huang, Wanda Hung, and Kang G. Shin, (2005) "Dynamic Data Replication in Free Disk Space for Improving Disk Performance and Energy Consumption", Symposium on Operating Systems Principles, pp263-276.

[17] Teorey, Toby J. and Pinkerton, Tad B, (1972) "A Comparative Analysis of Disk Scheduling Policies", Communication of the ACM, Vol. 15, No.3, pp177-184.

[18] Thomasian, Alexander and Lui, Chang, (2002) "Some new Disk Scheduling Policies and Their performance", Proceedings of ACM SIGMETRICS international conference on Measurement and modeling of computer systems, Vol. 30, Issue 1, pp266-267.

[19] Zoran Dimitrijevi, Raju Rangaswami, Edward Y. Chang, (2005) "Systems Support for Preemptive Disk Scheduling", IEEE Transactions on Computers, Vol. 54, No. 10,pp1314-1326.

[20] Mohammad Reza Bonyadi, (2010) "A genetic based disk scheduling method to decrease makespan and missed tasks", Information Systems Vol.35, pp791-803.

[21] M K Pakhira (2003) '"A Hybrid Genetic Algorithm using Probabilistic Selection", IE (I) Journal Vol.84, pp23-30.

## Authors

R.Muthu Selvi, Assistant Professor, Dept of IT has BE degree in Electronics and Communication Engineering from Madurai Kamaraj University in 1989. She secured the ME degree in Computer Science and Operating Systems, Multicore and Genetic Algorithms. She has published two papers in international journals and presented four papers in international conferences.

Dr. Rajaram Ramasamy, Professor& Head, Department of IT, Thiagarajar College of Engineering, has BE degree in Electrical and Electronics Engineering from Madras University in 1966. He secured the M Tech degree in Electrical Power Systems Engineering in 1971 from IIT Kharagpur and PhD degree on Energy Optimization from Madurai Kamaraj University in 1979. He and his research scholars have published more than 45 research papers in Journals and Conferences. Nine of his scholars secured the Ph.D. degree in Computer Science and Communication. His current areas of interest are Cryptography, Data mining