On Minimum Variance CPU-Scheduling Algorithm for Interactive Systems using Goal Programming

Anas Jebreen Atyeeh Husain
Information Systems Department,
Al al-Bayt University, Mafraq, Jordan

ABSTRACT

Improving response time is considered a fundamental objective in interactive environments. CPU scheduling aimed mainly to optimize the response time by minimizing its average in order to attain faster responses to users' requests. However, for interactive systems, reasonable and predictable services are more preferred than faster responses but highly variable. Delivering service in a timely manner at less variable response time is an issue that has been addressed in this paper. A goal programming (GP) model is proposed to perform CPU scheduling at minimum variance and low response time. The GP method determines the optimal process in the ready queue that best minimizes the variance to be executed first. A simulation system that can generate varied scheduling situations was developed and several tests were conducted. The performance of the proposed GP scheduling method is measured and compared to the other related scheduling methods. The evaluation results show that the GP scheduling method can provide predictable and reasonable service and it performs scheduling at minimum variance and lower response time. The GP method outperforms the other related methods with varying degrees.

Keywords

CPU scheduling, Goal programming, Interactive systems, Response time, Variance in response time

1. INTRODUCTION

Interactive computer systems such as timesharing and servers enable their users to communicate directly with the system; users give instructions and wait for a fast response which is considered a good service [1]. Responsive operations are desirable in such environment where timing violations decrease the provided quality of service (QoS) [2]. Accordingly, a method for ensuring reasonable response time is required [3]. Providing reasonable responses to users' requests is carried out by the operating system (OS) [4]. While swapping and using virtual memory can contribute to attain reasonable response time [3], CPU scheduling is fundamental in improving several desirable performance attributes such as response time [5]. CPU scheduling decides which process from among ready processes that are simultaneously available in the memory will run first [6].

One important goal that scheduling attempts to achieve in interactive systems is to improve the response time and run interactive applications timely [7]. Improving response time is an important and a primary objective adopted in the field of [8], [9], and in computer performance in general [10]. Improving response time has long been taken into consideration in the performance of interactive systems [11].

Providing low response time, and providing predictable and reasonable response time are two important performance attributes that can help optimize response time in interactive systems [12]. In optimal system design, the response time can be best optimized by achieving both performance attributes [13]. Minimizing the average of response time can help provide low response time [14], while minimizing the variance in the response time can help provide predictable and reasonable response time [1]. However, Response time has been mainly optimized by minimizing the average [15]-[21], [8] or the maximum response time [22]-[24] rather than the variance. Performing CPU scheduling at minimum variance is a concern that has not been addressed yet [1], [25], [26].

In fact, the variability or variance in the response time that is received by users can indicate predictable and reasonable response time [1], [12], [27], [28], and can be a fairness measure among the users [29]. That is, the less variance there is in the response time, the more reasonable and predictable response time and service users can get. Wierman and Harchol-Balter [25] denote that providing predictable response time has become an important requirement for many modern application designs. Silberschatz et al. [1] state in their operating systems text "A system with reasonable and predictable response time may be considered more desirable than a system that is faster on the average, but is highly variable." and "… for interactive systems, it is more important to minimize the variance in the response time than it is to minimize the average response time.". Usually, it is more preferred for users to receive reasonable and predictable response time than optimal response time on average [30]. The reason is that variable responses can frustrate users more than the large average response time can [12].

However, optimizing response time by minimizing measures other than variance in the response time may not necessarily indicate that all users get good service [13], [30], and cannot guarantee fairness or QoS among users [12], [31]. Some processes may get a priority at the expense of others [22] e.g. smaller processes may be preferred for execution over larger ones [25], [31]. Highly variable responses can disappoint users especially when they wait much longer than expected [27], [28]. Such consequences may lead to degrade reliability and trust in the system [32], and, in turn, this may cause lack of satisfaction among the users [33]. As a result, users may be forced to find alternative applications.

Variance in the response time can be a fundamental measure to optimize the response time that enhances the provided service, and is essential for applicability in modern computer systems [27], [28]. Hence, the focus in this paper on optimizing response time by minimizing the variance when performing CPU scheduling. The problem this paper tackles can be formulated in the following question:

- How can a CPU scheduling at minimum variance and low average response time be performed?

Consequently, a CPU scheduling method based on goal programming (GP) is proposed. The GP scheduling method can find and select optimal process in the ready queue that best minimizes the potential variance in the response time to be allocated the CPU. A detailed design of the proposed GP scheduling model is presented in section 3 that follows the related works in the next section. Evaluation results and discussion are introduced in section 4. Finally, some conclusions are given in section 5.

2. RELATED WORKS

Scheduling is an important OS function that can deliver a QoS [34]. Computer resources – including the CPU – need to be scheduled to achieve efficient use. A scheduling method can be selected based on the match between the requirements of the target environment and the goals of that method [3]. Scheduling has been mainly studied to minimize the turnaround time, waiting time, and response time, and to maximize CPU utilization, throughput [31], [34]. For instance, interactive systems are commonly used on personal computers and servers, besides optimizing response time is an important requirement and a primary goal in such environment [9], [19].

The objective of this paper is to perform a CPU scheduling that optimizes response time at minimum variance. Response time has been mainly optimized by minimizing the average [8], [16]-[21], [35], or the maximum response time [23], [24]. Average response time is a standard measure in computer performance [10] that has been analyzed under various scheduling strategies [36,55] and used to optimize response time in various environments such as: interactive-timesharing system [19], parallel computer system [10], [37], [38], cluster rendering system [17], web based requests [18], [20], [21], [39], bottleneck network link [40], and broadcast scheduling [41]. However, performing scheduling based on other criteria than variance in the response time may improve the system performance partially by providing fast responses for specific users rather than all other users in the system. Such a concern may result in lower level of QoS provided in the system and less satisfaction among users. That is, some users may wait longer than expected, some users may get priority over others arbitrarily, and the responses might be highly variable.

The variance in the response time is an important requirement to deliver reasonable and predictable service for users, especially in interactive environments [1], [12], [42]. Some studies revealed that delivering consistent service in a timely manner at less variable response is an issue and it still needs to be addressed [1], [3], [25], [26]. Other studies pointed out that the variance in the response time is an important measure to be considered in optimal system design [13]. The relation of the response time and its variance to the user satisfaction has been investigated and analyzed in several perspectives such as: the influence of response time on users' satisfaction [26], the predictability of response time [25], the user perception of computer system response time [13], and the distribution of response time for users [12], [43]. It can be concluded that response time plays an essential role in determining user satisfaction [54].

Several common scheduling methods that adopt varied selection criteria other than response time and variance have been used to optimize response time. First-come-first-served (FCFS) method schedules processes based on the order of arrival that is optimal in minimizing the maximum response time [44]. Shortest-job-first (SJF) and shortest-remaining-time-first (SRTF) select processes for execution based on the burst time and they are intended to minimize average response time [45]-[47]. A process with the highest predetermined priority is allowed to run first in priority scheduling methods. Lottery scheduling is another responsive method [48] that selects a process which holds lottery tickets for system resources. Such common scheduling methods and their performance are discussed by Tanenbaum [3], Qureshi [5], and Silberschatz [1].

However, there are no methods that adopt response time and its variance as selection criteria. Other criteria have been adopted with the aim of optimizing the response time which limits the improvement in response time. Additionally, methods that aimed to optimize response time mainly have minimized its average or maximum, and little attention has been given for CPU scheduling methods that minimize variance in the response time [1]. Alternatively, the use of the variance in the response time as selection criteria is proposed to be a reference for all scheduling decisions with the aim of minimizing the variance and lower the response time. Finally, the variance has been derived, analyzed, and confirmed its importance under some scheduling methods [36], [49], but improving variance in the response time for processes and their distribution has received little interest [25]. The response time that each process can receive will be measured, analyzed and improved for the whole system in the proposed solution.

3. THE MINIMUM VARIANCE GOAL PROGRAMING SCHEDULING METHOD

A user in the interactive systems requests a service by submitting a job or process, and the system schedule processes in some approach to be allocated the CPU. Response time represents the time between submitting a job and getting a response [42], [50], [51] that needs to be optimized in such environments. To provide a reasonable and predictable service for all users, our proposed solution is to perform minimum variance CPU scheduling. The proposed scheduling method is a GP model that makes all scheduling decisions based on the potential variance in response time that might be resulted when a process is selected for execution. The GP method determines optimal process in the ready queue that best minimizes the variance to be allocated the CPU. The variance in the response time can be represented by the squared distance or gap from the average of response time for all processes [44]. Such selection strategy can minimize the variance and maintains low response time in average for all processes.

Consequently, the gap or distance of the response time from the average, and the amount of time that the process has used the CPU are proposed as a selection criteria that guide minimum variance CPU scheduling. The GP model is responsible for finding an optimal process that best achieves these two criteria in order to be selected and allocated the CPU. Firstly, the distance between the response time and the average of response time is the difference between the two values and it can help maintain low variance by selecting a process that has larger distance when performing each selection. Normally the processes that have larger response time from the average wait longer without response than the other processes and it is preferred to select first such a process for execution. This prevents the system from ignoring the processes that wait longer which in turn minimizes the variance and maintains low response time. Secondly, the amount of time that the process has used the CPU is another selection criterion that can contribute to minimizing the variance by selecting processes that have used the CPU less. This selection strategy gives higher priority for processes that have not used the CPU yet, or that have used the CPU in lesser time. The amount of time that the process has used the CPU criterion is also useful for situations in which several processes have response time with equal distance to the average with varying degree of CPU usage. Additionally, this criterion can help maintain less waiting time that might be increased if the first criterion is only adopted.

However, a challenge lies in finding a solution that best achieves such criteria concurrently; an optimal process that has maximum value of response time to the average and minimum time of CPU usage may not exist. Moreover, a process that has the maximum value of response time to the average may not be the one that has minimum time of CPU usage, and vice versa. Various processes may satisfy each proposed selection criteria with varied degrees, and the optimal solution is a compromise. Selection criteria and their level of achievement must be taken into account concurrently.

Accordingly, a GP model responsible for finding an optimal process that best satisfies such multiple selection criteria to be executed is proposed. GP is a multi-criteria satisfying approach that seeks a solution that best fits or satisfies the related criteria in a multi criteria decision making (MCDM) problems [52]. GP scheduling model performs scheduling at minimum variance and low response time by (i) selecting a process that has the maximum distance between the response time and the average, (ii) selecting a process that has the minimum amount of CPU usage time. A set of related constraints is developed and used to help select an optimal process that best minimizes the response time and the variance. Finally, the following GP model is developed and constructed as presented in Figure 1:

Min $d_D^- + d_D^+ + d_T^- + d_T^+$;

Subject to:

$(\sum_{i=1}^{n} D_i * P_i) - d_D^- + d_D^+ = \text{Maximum (D)}$;

$(\sum_{i=1}^{n} T_i * P_i) - d_T^- + d_T^+ = 0$;

$(\sum_{i=0}^{n-1} P_i) = 1$;

Fig 1: The proposed GP scheduling method

The variable *n* is the number of processes in the ready queue, D_i is the distance or difference between the response time value of a process *i* and the average response time value of all processes, Maximum (D) represents the largest difference *D* that exists at selection time in the system. T_i is the amount of time that a process *i* has used the CPU at selection time where maximum value of T_i equals to the burst time of that process. P_i is a binary decision variables (0/1 variables), and $d_D^-, d_D^+, d_T^-, d_T^+$ are deviational variables. The D, T, and Maximum (D) are input variables for the GP model and can be obtained dynamically from the system at each time of selection. The expected output of the model is a vector of 0/1 values corresponds to each process decision variable (P_i) where one process will be assigned the value 1 to be selected and allocated the CPU. The process with 1 value is the optimal process whose selection can result in minimum variance in the response time.

4. EVALUATION RESULTS AND DISCUSSION

In order to measure the ability of the proposed minimum variance scheduling method to perform CPU scheduling with the least possible response time and variance, and to compare its performance with related methods, a simulation system was constructed and several tests were run. The system simulates a ready queue that contained several ready processes, and performs scheduling by selecting processes for execution, and calculates response time, waiting time and variance for all processes after the execution has finished. The simulation that was executed to generate varied scheduling situations consists of different numbers of submitted processes in the ready queue with varied burst time, arrival time, and priority values for each process.

Scheduling situations were generated randomly under three different numbers of processes in the ready queue, where *n* = 10, 50, and 100. The burst time that was chosen randomly from among three intervals i.e. 1 and 10, 25, and 60 milliseconds (ms) represents the size of each process. For each of the 9 combinations of *n* and burst time values, arrival time and priority values were generated randomly between the interval 0 and 10 where it is found from running the simulation several times that these variables do not have significant effect on the performance of scheduling methods. For each of the 9 scenarios, 300 scheduling situations were generated resulted in a total of 2700 situations that require CPU scheduling. All scenarios have been classified, summarized, and tested as depicted in Table 1 into two sets: Set (A) is to measure and compare the performance of scheduling methods in scheduling different numbers of processes, set (B) is to measure and compare the performance of scheduling methods in scheduling processes with different sizes.

Table 1. Simulation Scenarios

No	Number of processes		Size (burst time)	
	Test	**Value**	**Test**	**Value**
1	$(A)_{10}$	10	$(B)_{1\text{-}10}$	10
2		10	$(B)_{1\text{-}25}$	25
3		10	$(B)_{1\text{-}50}$	60
4	$(A)_{50}$	50	$(B)_{1\text{-}10}$	10
5		50	$(B)_{1\text{-}25}$	25
6		50	$(B)_{1\text{-}50}$	60
7	$(A)_{100}$	100	$(B)_{1\text{-}10}$	10
8		100	$(B)_{1\text{-}25}$	25
9		100	$(B)_{1\text{-}50}$	60

Consequently, for each generated situation that requires a CPU scheduling, the simulation is implemented and configured to schedule all processes in the ready queue using seven methods namely; the FCFS, the SJF, SRTF, the non-preemptive priority scheduling (N-Pr), the preemptive priority scheduling (P-Pr), minimum distance to average scheduling, and the proposed GP scheduling method. For each execution of the simulation, each method selects a process from the ready queue to be allocated the CPU until execution completes. Thereafter, the response time, the waiting time and the variance in response time for all processes have been computed and recorded for that method. However, using minimum distance to average scheduling method in the simulation is proposed to measure and compare its performance to the GP scheduling method and to show that adopting the first proposed selection criterion only as a solution is not enough to achieve minimum variance even that it is a simple method that doesn't require an MCDM technique.

The obtained evaluation results about response time, waiting time and the variance achieved by all methods have been analyzed and classified into several portions. Initially, the aim is to measure and compare the performance of scheduling methods and their ability to handle different environmental changes. Figure 2 presents the effect of varying environmental changes such as the number of processes in the ready queue and process size on the performance of scheduling methods, and the ability of scheduling methods to minimize response time, waiting time, and the variance in response time when performing CPU scheduling. The results show that the variance and response time that received by processes increase with the increase of both of the number and the size of processes. Additionally, the results show that the GP method significantly reduces the response time and variance, and slightly increases the waiting time over all method for all scenarios. Indeed, the proposed GP method can scale to varying environmental changes at faster responses and minimum variance which is considered as a good service for the users. The results prove that performing CPU scheduling based on other criteria than response time and variance may maintain better waiting time but it leads to a significant increase in response time and variance in the responses for users. Using GP scheduling method leads to increase of waiting time but this increase is not significant especially if compared to the significant improvement in the response time and the variance which is in turn more desirable in the interactive [1], [12], [42].

The expected variance and response time increase with the increase of the number of processes in the ready queue. For example, few processes that first arrive receive faster responses than the other large number of processes when the methods that perform scheduling based on arrival time have been used, or few processes that have short burst time receive shorter responses than the other large number of processes when the methods that perform scheduling based on burst time have been used, or few processes that have high priority receive shorter responses than the other processes when the methods that perform scheduling based on static priority have been used. As a result, the response time and the variance achieved in the system have been increased with the increase of the number of processes in the ready queue, and all other methods cannot scale to varying number of processes at fast responses and low variance.

Furthermore, the expected variance and response time increase with the increase of the size of processes in the system. Larger processes - especially interactive ones - stay longer in the ready queue and receive longer responses and higher variance than smaller processes when performing scheduling based on arrival time or static priority. Small processes leave the ready queue faster and the response time is relatively lower and so is the variance. Additionally, using scheduling methods that adopt the shortest burst time also make the selection of larger processes after executing all smaller processes last. As a result, the response time and the variance achieved in the system have been increased with the increase of the size of the processes in the ready queue, and all other methods cannot scale to varying size of processes at fast responses and low variance. To sum up, the increase of both of the number and the size of processes increases the challenge of performing CPU scheduling at low variance in response time and there is an immense need to be maintained.

Instead, the proposed GP method performs CPU scheduling for all situations and scenarios at minimum variance and low response time over the other methods. This could be explained as the GP selects processes based on the potential response time that might be resulted and based on its ability to minimize the variance. Furthermore, GP seeks processes that best achieve the proposed two main selection criteria simultaneously which have better result. Comparatively, scheduling method such as minimum distance do not take into account the amount of time that the process has used the CPU which increased the response time and the variance. Furthermore, other scheduling methods such as FCFS, SJF, SRTF, and priority do not take into account the response time or the variance in their selection strategies which significantly increases the response time and the variance that is considered one of the most important scheduling objective in interactive environments [53].

Secondly, it is aimed to analyze and compare the distribution of the responses that the users can receive when different scheduling methods are used. The simulation is executed for 25 processes with random size between 1 and 60 seconds. The processes are scheduled using all methods and the response time for each process in addition to the average of response time is computed, and the distribution of response time around their average is presented in Figure 3. The results show that the GP performs the best over all other methods and achieves lower average response time and better distribution around the average. That is, better distribution represents closer values of response time to their average and minimum variance in response time. This can be explained by the objective of the GP model which is to limit the increase of the distance by selecting processes that have larger distance from the average.

However, each time the process is selected from the ready queue, some processes may have a precedence or priority to be executed over the others according to the potential response time for each process at that time. This priority may dynamically change at other selection times with the change of the processes in the ready queue based on the new values of the arrival time, the burst time, the response time and the average of response time. Thus, when other methods ignore such a fact and adopt static priority, or adopt dynamic priority other than response time and variance, the response time increases in the average and in the variance.

When performing selection based on the arrival time, for example, the first processes will be repeatedly selected many times over other processes that have not used the CPU yet or that have used the CPU in a lesser time. The same issue is applied when performing selection from the ready queue based on burst time and static priority. Thus, using such scheduling methods increases the average response time and scatters the response time around their average. Comparatively, the GP performs its selection based on the potential response time that may be resulted at selection time which leads to a lower average response time and a better distribution around the average.

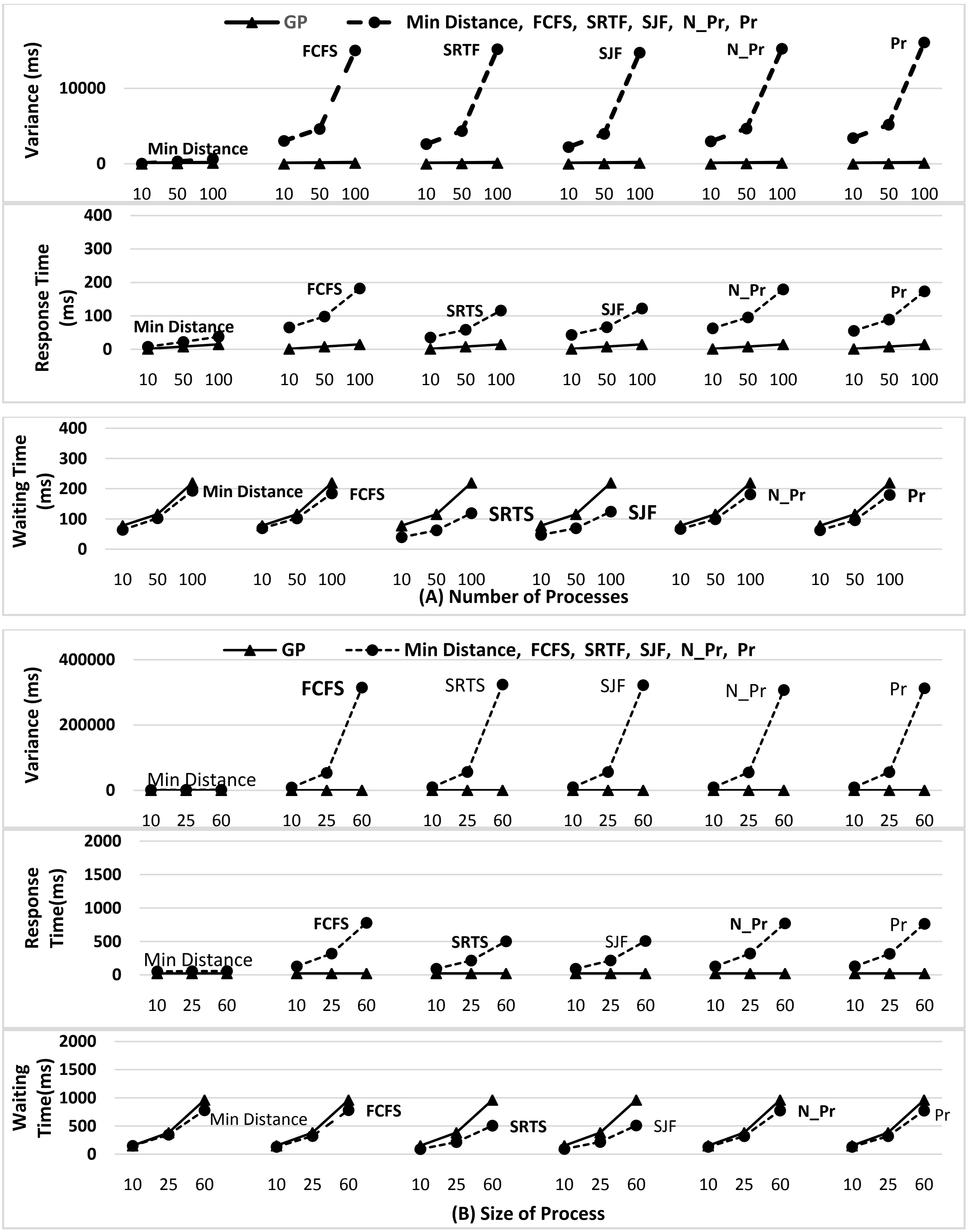

Fig 2: Performance of scheduling methods in scheduling (A) different numbers of processes (B) different size of processes.

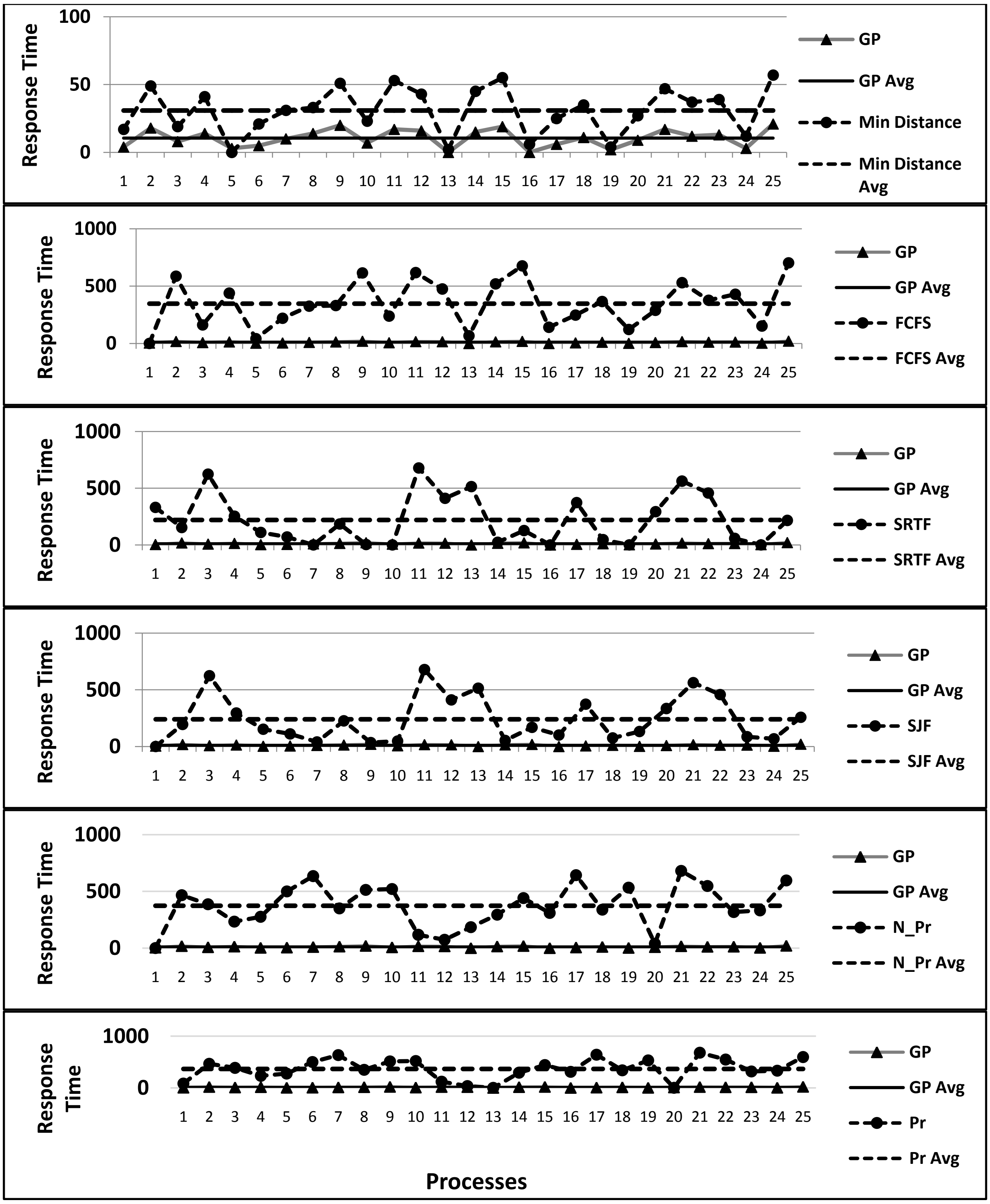

Fig 3: Distribution of the response time (in ms) received by each process around their average using different scheduling methods.

Table 2. Average performance of scheduling methods for all scenarios and their situations

	GP	Min. Distance	FCFS	SRTF	SJF	N_Pr	Pr
Variance (ms)	**299.5**	1504.6	125702.8	129940.3	129285.9	123788.7	125942.5
Response Time(ms)	**23.1**	55.7	408.2	268.1	271.3	406.0	401.7
Waiting Time (ms)	498.7	424.8	409.7	**269.8**	272.7	407.4	406.2

Lastly, the average performance of all methods for all scenarios is summarized and presented in Table 2. The minimum values are marked as bold indicating best achieved values. The GP method performs better than the other methods and completes CPU scheduling at minimum response time and variance for all generated situations and scenarios. The GP outperforms the other methods with varying degrees.

5. CONCLUSION

CPU scheduling at minimum variance in the response time has been addressed in this paper. The proposed solution is to find and select a process that best reduces the potential variance in the response time for execution. The distance of the response time from the average, and the amount of time that the process has used the CPU are proposed as a selection criteria that guide minimum variance CPU scheduling. Accordingly, a linear goal programming model is proposed to determine the processes that best satisfy the proposed criteria.

The performance of the proposed GP scheduling method has been measured and compared to related scheduling methods. The results show that the GP method performs better and significantly reduces the response time and variance. The proposed GP method can scale to varying environmental changes at faster responses and minimum variance. The GP outperforms the other selection methods with varying degrees. However, as a future development, another selection criteria might be proposed and used when performing CPU scheduling in order to best optimize other CPU scheduling criteria such as waiting time.

6. REFERENCES

[1] A. Silberschatz, P. B. Galvin, and G. Gagne, Operating system concepts. (John Wiley & Sons, 2013).

[2] S. Kato, Y. Ishikawa, and R. Rajkumar, CPU scheduling and memory management for interactive real-time applications. Real-Time Systems, Vol. 47, n. 5, pp. 454–488, 2011.

[3] A. S. Tanenbaum, and H. Bos, Modern operating systems. (Prentice Hall Press, 2014).

[4] M. A. Al-Husainy, Best-job-first CPU scheduling algorithm. Information Technology Journal Vol. 6, n. 2, pp. 288-293, 2007.

[5] I. Qureshi, 2014. CPU Scheduling Algorithms: A Survey. International Journal of Advanced Networking and Applications, 5(04): 1968-1973.

[6] S. Almakdi, M. Aleisa, and M. Alshehri, Simulation and Performance Evaluation of CPU Scheduling Algorithms. International Journal of Advanced Research in Computer and Communication Engineering, Vol. 4, n. 3, pp. 1-6, 2015.

[7] Y. Etsion, D. Tsafrir, and D. G. Feitelson, Process prioritization using output production: scheduling for multimedia. ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM), Vol. 2, n. 4, pp. 318-342, 2006.

[8] U. Schwiegelshohn, Preemptive weighted completion time scheduling of parallel jobs. SIAM Journal on Computing, Vol. 33, n. 6, pp. 1280-1308, 2004.

[9] Ramamritham, K., and Stankovic, J., Scheduling algorithms and operating systems support for real-time systems. Proceedings of the IEEE, (Page: 55-67 year of publication: 1994).

[10] Turek, J., Ludwig, W., Wolf, J. L., Fleischer, L., Tiwari, P., Glasgow, J.,and Yu, P. S., 1994. Scheduling parallelizable tasks to minimize average response time. Proceedings of the sixth annual ACM symposium on Parallel algorithms and architectures, PP: 200-209.

[11] Cotton, I. W., 1977. Cost-benefit analysis of interactive systems. Computer Networks, 1(6), 311-324.

[12] A. Wierman, 2007. Scheduling for today's computer systems: Bridging theory and practice. Unpublished dissertation in partial fulfillment of the requirements for the degree of PhD, Carnegie Mellon University, Pittsburgh.

[13] R. Geist, R. Allen, and R. Nowaczyk, Towards a model of user perception of computer systems response time. ACM SIGCHI Bulletin, Vol. 17, SI, pp. 249-253, 1986.

[14] Im, S., Kulkarni, J., and Moseley, B., Temporal Fairness of Round Robin: Competitive Analysis for Lk-norms of Flow Time. Proceedings of the 27th ACM on Symposium on Parallelism in Algorithms and Architectures, (Page: 155-160 year of publication: 2015).

[15] R. Krishnaswamy, Broadcast Scheduling: Minimizing Average Response Time. Encyclopedia of Algorithms, (Reference Work Entry), pp: 1-5.

[16] R. A. Kulkarni, and S. H., Patil, A survey on improving performance of Real Time Scheduling for Cloud Systems. International Journal for Innovative Research in Science and Technology, Vol. 1, n. 7, pp. 171-173, 2015.

[17] Li, Q., Wu, W., Zhou, X., Sun, Z., and Huang, J., R-FirstFit: A Reservation Based First Fit Priority Job Scheduling Strategy and Its Application for Rendering. Proceedings of the IEEE 17th International Conference on Computational Science and Engineering, CSE, (page: 1078-1085, 2014).

[18] G. You, X. Wang, and Y. Zhao, A Dynamic Requests Scheduling Model Based on Prediction in Multi-core Web Server. In: Internet of Vehicles – Technologies and Services, Lecture Notes in Computer Science, pp: 304-312, 2014. ISBN: 978-3-319-11166-7. DOI: 10.1007/978-3-319-11167-4_30

[19] S. M. Mostafa, and S. Kusakabe, Towards Minimizing Processes Response Time in Interactive Systems. International Journal of Computer Science and Information Technology Research, IJCSITR, Vol. 1, n. 1, pp. 65-73, 2013.

[20] G. You, and Y. Zhao, A weighted-fair-queuing, WFQ-based dynamic request scheduling approach in a multi-core system. Future Generation Computer Systems, Vol. 28, n. 7, pp. 1110-1120, 2012.

[21] Zhang, S., Wu, H., Wang, W., Yang, B., Liu, P., and Vasilakos, A. V., 2011. Distributed workload and response time management for web applications. Proceedings of the 7th International Conference on Network and Services Management, (page: 198-206 year of publication: 2011).

[22] C. Chekuri, S. Im, and B. Moseley, Online Scheduling to Minimize Maximum Response Time and Maximum Delay Factor. Theory of Computing, Vol. 8, n. 1, pp. 165-195, 2012.

[23] J. Chang, Online Broadcast Scheduling: Minimizing the Maximum Response Time. Theory of computing, Vol. 8, SI, pp. 165-195, 2012.

[24] C. Chekuri, S. Im, & B. Moseley, Minimizing maximum response time and delay factor in broadcast scheduling. In: Algorithms - ESA, Lecture Notes in Computer Science, pp: 444-455. ISBN: 978-3-642-04127-3.

[25] A. Wierman, and M. Harchol-Balter, Classifying scheduling policies with respect to higher moments of conditional response time. ACM SIGMETRICS Performance Evaluation Review, Vol. 33, n. 1, pp. 229-240, 2005.

[26] Hoxmeier, J. A., and DiCesare, C., System response time and user satisfaction: An experimental study of browser-based applications. Proceedings of AMCIS, (Page: 347 year of publication:2000).

[27] B. G. Dellaert, and B. E. Kahn, How tolerable is delay?: Consumers’ evaluations of internet web sites after waiting. Journal of interactive marketing, Vol. 13, n. 1, pp. 41-54, 1999.

[28] M. K. Hui, and L. Zhou, How Does Waiting Duration Information Influence Customers' Reactions to Waiting for Services. Journal of Applied Social Psychology, Vol. 26, n. 19, pp. 1702-1717, 1996.

[29] D. Raz, B. Avi-Itzhak, & H. Levy, Locality of reference and the use of sojourn time variance for measuring queue unfairness. SIGMETRICS Perform. Eval. Rev., Vol. 33, n. 2, pp. 39–41, 2005.

[30] S. Im, 2012. Online scheduling algorithms for average flow time and its variants. Unpublished dissertation in partial fulfillment of the requirements for the degree of Doctor of Philosophy, University of Illinois, Urbana-Champaign.

[31] A. Wierman, Fairness and scheduling in single server queues. Surveys in Operations Research and Management Science, Vol. 16, n. 1, pp. 39-48, 2011.

[32] D. F. Galletta, R. Henry, S. McCoy, and P. Polak, Web site delays: How tolerant are users?. Journal of the Association for Information Systems, Vol. 5, n. 1, pp. 1-28, 2004.

[33] B. Shneiderman, Designing the user interface: strategies for effective human-computer interaction. Applied Ergonomics. Vol. 24, n. 4, pp. 295, 2003.

[34] Chandio, A. A., Xu, C. Z., Tziritas, N., Bilal, K., and Khan, S. U., A comparative study of job scheduling strategies in large-scale parallel computational systems. Proceedings of the 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, IEEE, (page: 949-957 year of publication: 2013).

[35] R. Krishnaswamy, Broadcast Scheduling: Minimizing Average Response Time. Encyclopedia of Algorithms, (Reference Work Entry), pp: 1-5.

[36] Gupta, A., Im, S., Krishnaswamy, R., Moseley, B., and Pruhs, K., Scheduling jobs with varying parallelizability to reduce variance. Proceedings of the 22nd ACM Symposium on Parallelism in Algorithms and Architectures - SPAA ’10, (page: 11-20 year of publication: 2010)

[37] U. Schwiegelshohn, W. Ludwig, J. L. Wolf, J. Turek, and P. S. Yu, Smart SMART bounds for weighted response time scheduling. SIAM Journal on Computing, Vol. 28, n. 1, pp. 237-253, 1998.

[38] J. Edmonds, Scheduling in the dark. Theoretical Computer Science, Vol. 235, n. 1, pp. 109-141, 1999.

[39] L. Cherkasova, Scheduling strategy to improve response time for web applications. In: High-Performance Computing and Networking, Lecture Notes in Computer Science, Springer Berlin Heidelberg. pp: 305-314, 1998. ISBN: 978-3-540-64443-9.

[40] I. A. Rai, G. Urvoy-Keller, & E. W. Biersack, Analysis of LAS scheduling for job size distributions with high variance. ACM SIGMETRICS Performance Evaluation Review, Vol. 31, n. 1, pp. 218-228, 2003.

[41] R. Gandhi, S. Khuller, Y. A. Kim, and Y. C. J. Wan, Algorithms for minimizing response time in broadcast scheduling. Algorithmica, Vol. 38, n. 4, pp. 597-608, 2004.

[42] M. Ubale, and M. Rahaman,. Improving the Performance of CPU Scheduling in Interactive Systems. International Journal of Advanced Research in Computer Science, Vol. 4, n. 1, pp. 25-28, 2013.

[43] P. M. Broadwell, Response time as a performability metric for online services. Computer Science Division, University of California. 2004.

[44] N. Bansal, and K. R. Pruhs, 2010. Server Scheduling to Balance Priorities, Fairness, and Average Quality of Service. SIAM Journal on Computing, Vol. 39, n. 7, pp. 3311–3335, 2010.

[45] R. W. Conway, W. L. Maxwell, and L. W. Miller, Theory of scheduling. (Courier Corporation 2012). 978-1306365451.

[46] M. Harchol-Balter, B. Schroeder, N. Bansal, and M. Agrawal, Size-based scheduling to improve web performance. ACM Transactions on Computer Systems, Vol. 21, n. 2, pp. 207-233, 2003.

[47] N. Bansal, and M. Harchol-Balter, Analysis of SRPT scheduling: Investigating unfairness. ACM, Vol. 29, n. 1, pp. 279-290, 2001.

[48] Waldspurger, C. A., and Weihl, W. E., Lottery scheduling: Flexible proportional-share resource management. of the 1st USENIX conference on Operating Systems Design and Implementation, USENIX Association, (pp: 1-11 year of publication: 1994).

[49] S. F. Yashkov, Mathematical problems in the theory of shared-processor systems. Journal of Soviet mathematics, Vol. 58, n. 2, pp. 101-147, 1992.

[50] P. Singh, A. Pandey, and A. Mekonnen,,. Varying Response Ratio Priority: A Preemptive CPU Scheduling Algorithm, VRRP. Journal of Computer and Communications, Vol. 3, n. 4, pp. 40-51, 2015.

[51] N. kumar, and Nirvikar, Performance improvement using CPU Scheduling Algorithm. International Journal of Emerging Trends of Technology in Computer Science, Vol. 2, n. 2, pp. 110-113, 2013. Retrieved from http://www.ijettcs.org/V2I2.html

[52] A. J. Husain, New Roll-Up Operator for Non-Additive Numeric Measure Summarization. Contemporary Engineering Sciences, Vol. 6, n. 8, pp. 393 – 402, 2013.

[53] Kiran, R. S., Babu, P. V., and Krishna, B. M., Optimizing CPU scheduling for real time applications using mean-difference round robin (MDRR) algorithm. Proceedings of the 48th Annual Convention of Computer Society of India, Springer, (page: 713-721 year of publication: 2014).

[54] R. S. Chang, J. S. Chang, and P. S. Lin, An ant algorithm for balanced job scheduling in grids. Future Generation Computer Systems, Vol. 25, n. 1, pp. 20-27, 2009.

[55] V. Gupta, M. Burroughs, & M. Harchol-Balter, Analysis of scheduling policies under correlated job sizes. Performance Evaluation, Vol. 67, n. 11, pp. 996-1013, 2010.