International Journal of Engineering Trends and Technology
ISSN: 2231 – 5381 /doi:10.14445/22315381/IJETT-V69I3P219
Volume 69 Issue 3, 118-126, March 2021

A Real And Accurate Real-Time Task Scheduling Architecture With Self-Feedback X-Boost Machine Learning Technique

G.Kasi Reddy[1], Dr. D Sravan Kumar[2]

[1]*Research Scholar, CSE, JNTU Hyderabad, India,*
[2]*Principal , Professor Dept. of CSE, SR International Institute of Technology, Hyderabad, India,*

gkasireddy@rediffmail.com, dasojusravan@gmail.com

Abstract:

The real-time performance analysis of software applications such as multiprocessors, uni-processors, and real-time operating systems can increase their demand. A Hard-Real-Time embedded, satellite, industrial, telecommunication, and robotics systems are widely used in RTOS applications. In these, applications, their performance is mainly depending on-time deadline, logical outcomes, the correctness of relies upon, and accuracy. Many scheduling algorithms are designed, but those are specifically workout on available deadlines as well as facing the above limitations. Therefore an advanced real-time task scheduling in the operating system is necessary to crossover the limitations and for improvement. In this research work, an advanced X-boosting machine learning technique is proposed for real-time task scheduling multi programming purposes. This work is providing fault-free, attribute, deadline, and un-restricted execution at final calculating the performance measures like as fault detection, sensitivity, worst-case performance ratio, accuracy, Arrival Time, Compilation Time, Deadline time Energy and f1-score. This self-feedback X-boosting machine-learning technique is outperformance the methodology and competes with present technology.

Keyword: *task scheduling, real-time programming, compilation time, and machine learning*

I. Introduction

The scheduling algorithms are more reliable in nature and perform every job perfectly with compilation time and rely on time and required energy. The main focusing of this work can diminish the operating speed as well as available resource utilization. The major parameters like the earliest deadline first (EADF) is an important entity that can sort out the real-time schedules. In earlier online scheduling algorithms are implemented, but those are facing problems such as processor demand and task execution. Therefore crossover the following limitations are a major task to improves the exact feasible test. In the coming decades, these scheduling algorithms are improved and optimized for future applications [1].

Real-time scheduling algorithms are mainly focusing on primitive entities and scheduling tasks. These real-time applications are not continuously energized the scheduling tasks. Many research proposals have proven that the EADF algorithm is most suitable for periodic tasks as well as deadline times [2, 12-15]. The extension of entities and primitive problems regarding multiprocessors can improve the scheduling algorithm range. In software engineering, parameters like compiling time, scheduling time, and deadline parameters are most critical with non-primitive real-time task algorithms.

Usually, a self-efficacy method such as the RM (Rate Monotonic) equation schedules the processes in which repetitive operations are conducted. The importance of the mission in this method is determined by the process rate (period). The dynamic scheduling routing protocol is used by the EDFF method in which the tasks normally change (EADF). The role is focused on the importance of the task.

In the last decades, artificial intelligence has been a hot subject in the field of research into computer vision. In several areas, such as voice commands, machine learning, multimedia analysis, image processing, and graphics, it has created a bunch of well-being in specific applications. The pace of technology is changing, particularly the proliferation of government computer applications.

Literature survey

S NO	AUTHOR	TECHNIQUE	KEY POINT	ADVANCED MODEL
1	S. Baruah et al. [2013]	EDF scheduling on similar multiprocessor platforms.	This method introduced the splitting scheme for EDF identical multiprocessors.	Deep learning
2	M. Cheramy et al. [2013]	A technique based on overheads.	This method discussed about the study of real-time scheduling.	Machine learning
3	Hladik et.al [2014]	SIMSO offers an easy way to generate tasks.	In this paper, we discuss about Comparison of numerous scheduling policies.	Neural networks
4	Erickson et.al [2013]	A technique based on a global fair lateness algorithm.	This paper discussed about G-FL is better than G-EDF for SRT systems.	Deep learning
5	Goossens et.al [2017]	It is based on cyber-physical systems	In this paper, it provides appropriate modules and attributes.	Machine learning
6	Lindh et.al [2010]	A technique based on scheduling algorithms.	This method introduced the extension algorithms protocols.	Machine learning
7	Rouhifar et.al [2015]	A technique based on Hard RTS.	In this paper, we discuss about the schedulability and Qos factors.	Neural learning
8	Saranya et.al[2015]	It is based on DP scheduling algorithm	It based on Real-time Virtualization System Activities	Deep learning
9	Salam et.al [2019]	It is based on ACO (ant colony optimization).	In this paper, we compared EDF, ACO, and GA algorithm	Neural learning
10	M. Chetto et al. [2014]	A technique based on energy harvesting.	In this paper, we discuss about the real-time energy storage device.	Machine learning
11	Zhou. J et.al [2020]	This work about Complex embedded systems.	It performs the flexible transition mode	Deep learning
12	Salgado R M et al. [2018]	It is based on a two-phase flow.	We discuss about the correct pattern in terms of the flow of the section	Deep learning
13	Bertozzi M et.al [2018]	It is based on AVL (automatic vehicle localization).	In this paper, we discuss about the capability of improving the quality of life visually.	Neural learning
14	Ebert C et al. [2019]	A technique based on technological innovation.	This method introduced about the connecting competitiveness.	Machine learning
15	Huang X et.al [2018]	Tangential relative displacements.	We discuss about the configuration of the damping ring.	Neural learning
16	Zheng W et.al [2017]	A technique based on cache allocation	This method introduces about minimizing the number of task preemptions	Deep learning
17	Ghobaei Arani et.al [2020]	Cyber-physical system	In this paper, we discuss about task scheduling as the NP	Machine learning
18	Huang et al. [2020]	A technique based on cps prediction accuracy	This method introduces about task optimization and scheduling algorithm.	Deep learning
19	Karimi M et.al [2020]	It is based on intermittently powered devices	In this paper, we discuss about timekeeping in the presence of intermittent power losses.	Deep learning
20	Casini D et.al [2020]	A technique based on predictability	This method is based on the execution of DNNS with real-time tasks.	Neural learning

Methodology

In this research work, a machine learning-based real-time x-boosting scheduling algorithm is used on an RTOS processor to meet the compiling time. This is a machine learning technique consist of a processing module and task scheduling module, and these are used to balancing the optimal performance. In this, the real-time task is taken as an example, such as arrival time, compilation time deadline, and energy rate. These four parameters are defined with the periodic task as well as independent in nature. The following parameters are performing tasks named as {T1, T2……..TN}. The first parameter is performing arriving task representing Ti, arriving time denoted as Ai, coming to the second one, i.e., execution time represented as Ci, the deadline time denoted as Di, and final energy consumption time represented as Ei. The Ei and Ci are the fully independent parameters that cannot have any relation between the task like execution time and energy consumption.

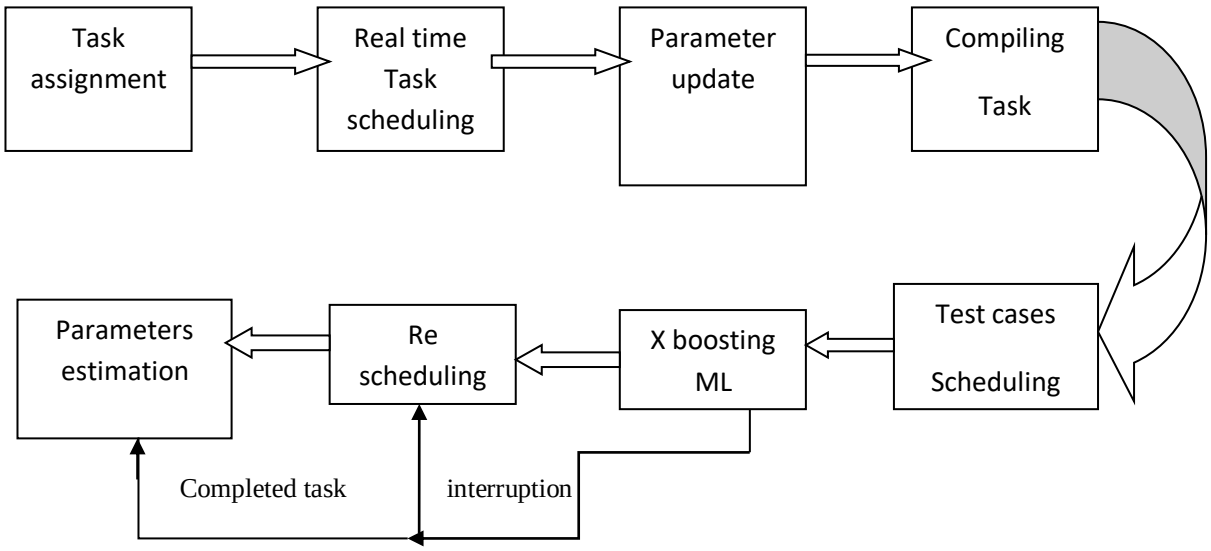


Figure: 2. Proposed X boosting ML

Algorithm: X-boosting Real-time task scheduling

Input: Real-time task on RTOS

Step 1: Initialize the parameter classification

Step 2: task entity identification

Step 3: validate the X boosting classifier using the following mathematical analysis

F_0 (x) = arg_γ min$\sum_{i=1}^{n} T(A_i, Ci)$ ----- (1)

The above equation clearly explains about Fitness function of the X boosting model, T is a task, A and C are the arrival as well as compiling time, respectively.

$D = -\left[\frac{\partial T(C_i, F(x_i))}{\partial F}\right]_{F(x)=F_{m-1}(x)}$ for i=1… n--- (2)

Equation 2 clearly explains about deadline time calculation using a fitness function with a particular task number.

$\{(C_i, D_{im})\}_{i=1}^{n}$

$C_m = arg_\gamma \min \sum_{i=1}^{n} T(y_i, F_{m-1}(x_i) + Ch_m(x_i))$. ----(3)

The above equation 3 explains about compilation time estimation using argument function, here C&D inverse rank functions if this value is 1, the deadline and compilation time can perform within the expected time. Otherwise, the error element is shooting out the real-time task scheduling.

$E_m(x) = F_{m-1}(x) + h_m C_m(x)$ ----(4)

Equation 4 describes about energy calculation for overall task completion; this model can help to find out the task scheduling functionality.

$h_M(x) = \sum_{j=1}^{j_m} b_{jm}\ 1_{R_{jm}}$ (x) ----- (5)

The above equation 5 estimating all performance metrics like as arrival time, deadline, compilation time, and task energy. Here is the task impulse response function, and b is the particular tree in the specified class.

$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x), \gamma_m =$
$arg_\gamma \min \sum_{i=1}^{n} T(y_i, F_{m-1}(x_i) + Ch_m(x_i))$ ---(6)

$F_m(x) = F_{m-1}(x) + \sum_{j=1}^{j_m} b_{jm} 1_{R_{jm}}(x), \gamma_m = arg_\gamma \min \sum_{x_i \in R_{jm}} T(y_i, F_{m-1}(x_i) + \gamma)$ --- (7)

The above equation 6&7 demonstrates that x-boosting machine learning function with local classes, it can provide the test features from available fitness function.

Step: 4 identify the compilation time, deadline, and arrival time as well as energy consumption.

Step: 5 stop the process

Mathematical modeling of scheduling mechanism

Below, equation 8 explains about n periodic tasks attained by the proposed machine learning model.

$$U = \sum_{i=1}^{n} \frac{Com_i}{Time_i} \leq n\left(2^{\frac{1}{n}} - 1\right) \quad \text{------ (8)}$$

In this computation, com is a computation time and time representing a task release period, in particular RTOS.

$$L_i(t) = D_i(t) - E_i(t) \quad \text{------ (9)}$$

Here L is a lack of effects in real-time task maintenance, D represents that deadline time, as well as E, is an execution task time.

$$\forall_i, p_i = D_i - \frac{m-1}{m} C_i \quad \text{------ (10)}$$

P_i is a priority task, and D_i is a Deadline, C_i is an execution time. The following parameters have to provide the estimation of task scheduling.

$$l_{i,0} = U_i(t_{f1} - t_{f0}) \quad \text{---------- (11)}$$

The above equations clearly explanations about pending local execution time

$$\log(T, t) = wt(T).t - \sum_{u=0}^{t-1} S(T, u) \quad \text{------ (12)}$$

Wt (T) is an allotted task in T slot, t is a set of tasks, $S(T, u)$ =1 task schedule in slot t.

$\forall T, t: -1 < lag(T, t) < 1$: it is a length of subtasks

$$r(T_i) = \left|\frac{i-1}{wt(T)}\right|, \quad d(T_i) = \left|\frac{i}{wt(T)}\right| \quad \text{------ (13)}$$

release time and pseudo deadline time

Results

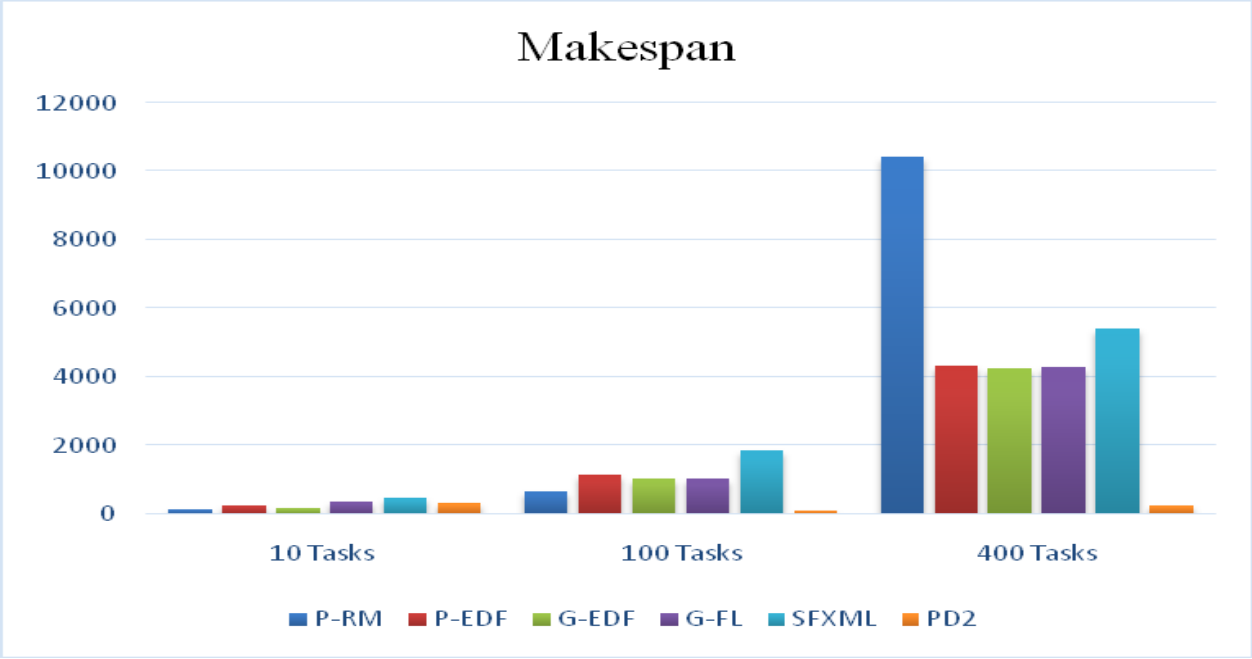


Figure: 3 Makespan time

	P-RM	P-EDF	G-EDF	G-FL	SFXML	PD^2
10 Tasks	148.74	274.316	169.246	386.464	499.782	321.628
100 Tasks	668.551	1141.018	1028.735	1037.129	1878.292	117.907
400 Tasks	10435.35	4349.314	4251.912	4284.356	5436.178	266.433

The above figure clearly explains about Makespan time analysis; here, various models are performed like P-EDF, G-EDF, G-FL. The comparison analysis is performed on 10tasks, 100 tasks, and 400 tasks; in this, all conditions proposed model getting more improvement compared to real techniques.

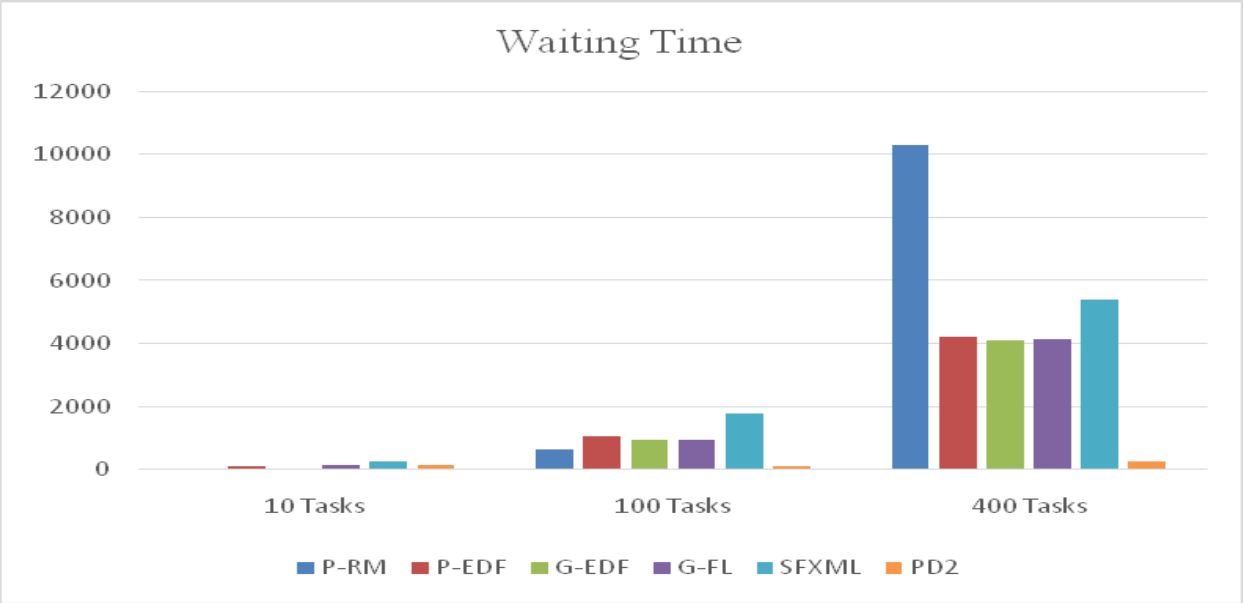


Figure: 4 waiting time analysis

	P-RM	P-EDF	G-EDF	G-FL	SFXML	PD^2
10 Tasks	16.514	132.407	27.337	150.628	263.946	140.2
100 Tasks	608.511	1060.68	948.396	956.79	1797.953	114.1
400 Tasks	10284	4199.21	4096.08	4128.53	5390.231	258.992

Figure 4 describes about waiting for time analysis of various task scheduling algorithms. It is clearly identified that the partitioned EDF, partitioned RM, GEDF, GFL models are getting some latency from high-density tasks. Moreover, our proposed SFXML model getting more improvement and giving the accurate results

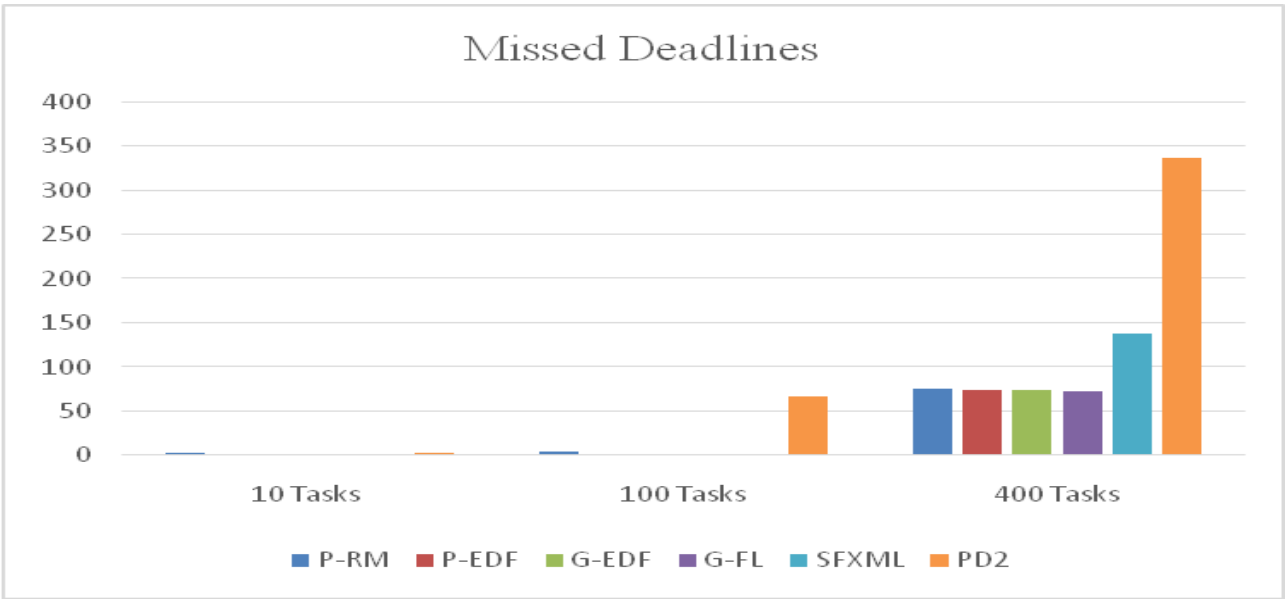


Figure: 5 Missed Deadline time analyses

	P-RM	P-EDF	G-EDF	G-FL	SFXML	PD^2
10 Tasks	3	1	1	0	0	3
100 Tasks	5	0	0	0	0	67
400 Tasks	75	73	73	73	137	336

Figure 5 explains about missed deadline time of various models such as partitioned EDF, partitioned RM, GEDF, GFL models. In this, the proposed SFXML model attains more improvement at 10tasks, 100 tasks, and 400tasks situation. 2

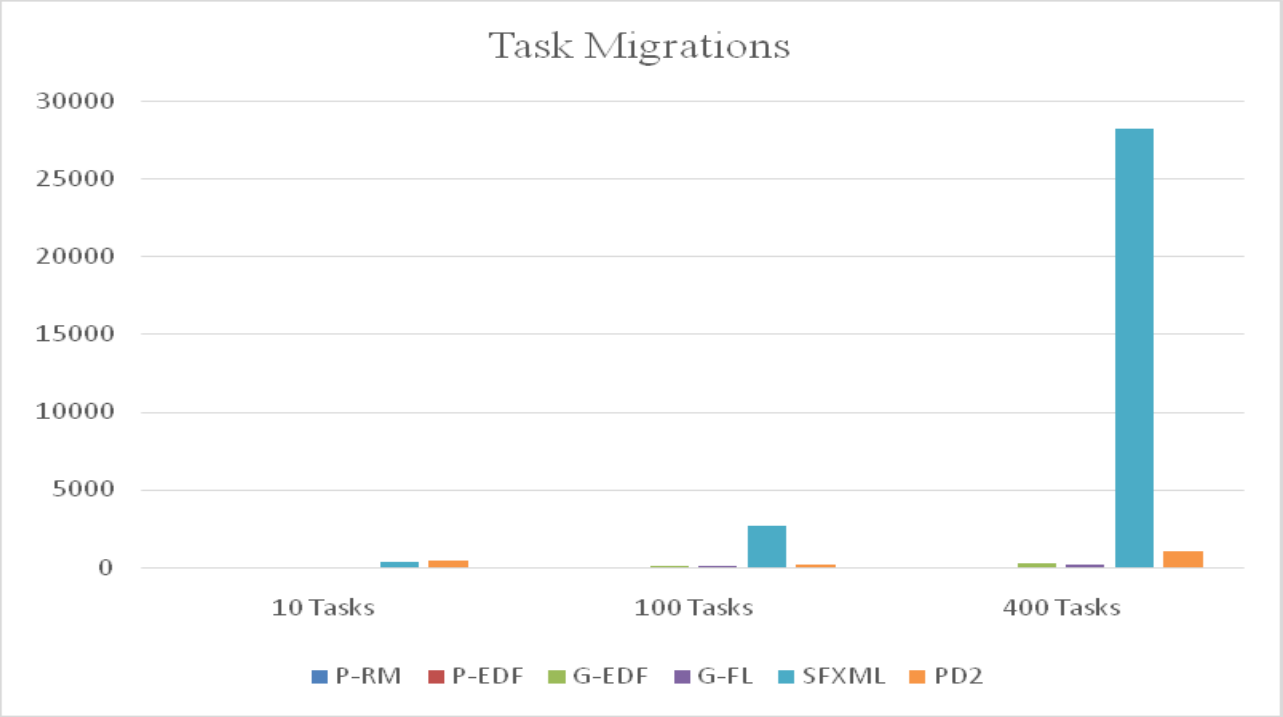


Figure: 6 Task migrations

	P-RM	P-EDF	G-EDF	G-FL	SFXML	PD^2
10 Tasks	0	0	0	9	320	425
100 Tasks	0	0	80.25	81.844	2671.234	152.446
400 Tasks	0	0	216.002	216.814	28156.11	995.5

Figure 6 explains about task migration element analysis. In this, in any situation, the proposed SFXML model getting a high migration score compared to earlier models, Such as partitioned EDF, partitioned RM, GEDF, GFL models.

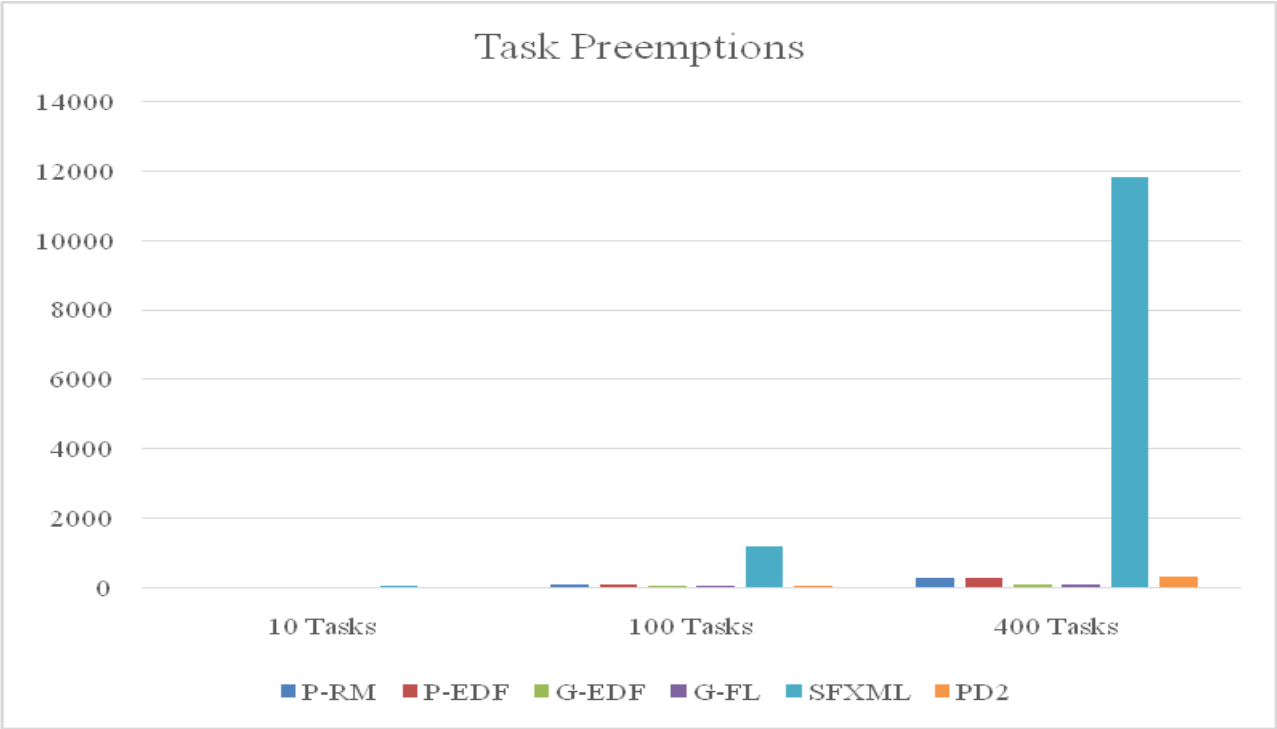


Figure: 7 task permission analysis

	P-RM	P-EDF	G-EDF	G-FL	SFXML	PD^2
10 Tasks	3	5.516	1	3	27	13
100 Tasks	84.52	94.007	30.105	23.481	1144.474	56.48
400 Tasks	267.064	266.045	74.234	71.536	11803.85	303.5

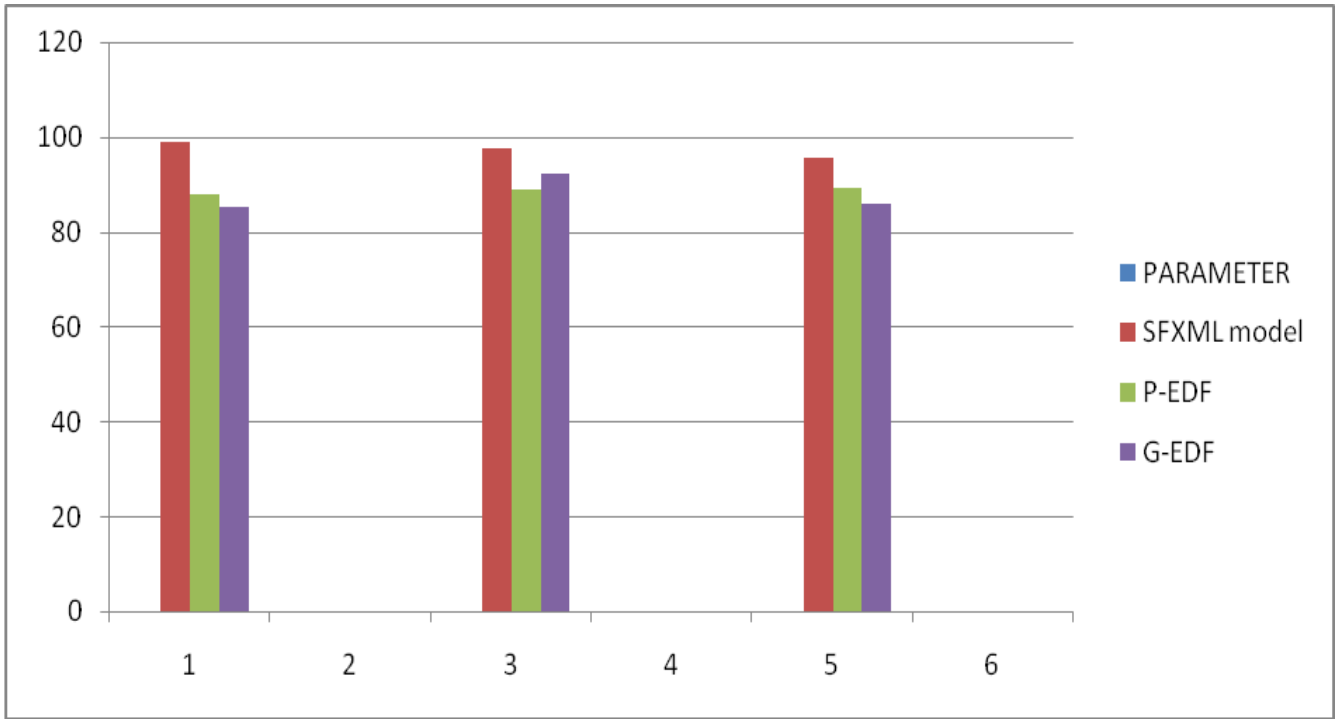
Figure 7 describes task permission analysis of the proposed SFXML scheduling algorithm; in this analysis, all earlier models getting less accurate results. Therefore at 10tasks, 100tasks and 400tasks condition our task scheduling technique getting high-end outputs

$$\text{Sensitivity} = Tp/Tp + Fp$$

$$\text{Accuracy} = (TP + TN) / (TP + TN + FP + FN)$$

$$\text{Recall} = Tp/Tp + Fn$$

S NO	PARAMETER	SFXML model	P-EDF	G-EDF
1	X- boosting Prediction Probability	[[3.5565485e-04 9.79614826e-01]]	[3.12e-4 1.3987]	[4.12e-4 1.3987]
2	Accuracy	98.83	87.81	85.27
3	sensitivity	97.45	88.87	92.28
4	Recall	95.46	89.12	85.81



The above table and figure 8 explain about performance measures of the SFXML model. In this, the accuracy, sensitivity, and recall scores are generated using a confusion matrix. In all conditions, our proposed model is getting more improvement, i.e., accuracy 98.3%, sensitivity 97.45%, and recall 95.46%; this is a high improvement and competes with existed model.

Conclusion

In this article, we provided an analysis of X boosting machine learning algorithms for real-time scheduling that are categorized into algorithms for RTOS and multiprocessor scheduling. We then made a distinction with the real-time scheduling algorithms of the multiprocessors that are also categorized into partitioned and global scheduling algorithms. These findings reveal that the divided analysis techniques in certain parameters that are waiting time, missing deadlines, and work preemptions are better developed than X boosting scheduling algorithms. The global scheduling algorithms were higher in efficiency as the number of tasks grew than partitioned task scheduling except for two of them, with a large number of missed deadlines.

References

[1] Baruah, S., Partitioned EDF scheduling: a closer look. Real-Time Systems, 49(6) (2013) 715-729.

[2] Chéramy, M., Déplanche, A.-M. & Hladik, P.-E., Simulation of Real-Time Multiprocessor Scheduling with Overheads. Reykjavik, Iceland, hal-00815502 , (2013) 5-14.

[3] Chéramy, M., Hladik, P. -E. & Déplanche, A. -M., SimSo: A Simulation Tool to Evaluate Real-Time Multiprocessor Scheduling Algorithms. In Proceedings of the 5th International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS), (2014) 37-42.

[4] Erickson, J. P., Anderson, J. H. & Ward, B. C., Fair lateness scheduling: reducing maximum lateness. Real-Time Systems, 50(1)(2013) 5-47.

[5] Lindh, F., Otnes, T. & Wennerstrom, J., Scheduling Algorithms for Real-Time Systems., (2010).

[6] Goossens, J. & Richard, P., Multiprocessor Real-Time Scheduling. Wang X. (eds) Cyber-Physical Systems: A Reference, (2017) 1-33.

[7] Rouhifar, M. & Ravanmehr, R., A Survey on Scheduling Approaches for Hard Real-Time Systems, International Journal of Computer Applications, 131(17) (2015).

[8] Saranya, N. & Hansdah, R., Dynamic Partitioning Based Scheduling of Real-Time Tasks in Multicore Processors, Auckland, New Zealand, IEEE 18th International Symposium on Real-Time Distributed Computing, (2015).

[9] Salam, Abdul, Sohail Abbas, Yousaf Khan, Sanaul Haq, and Saeed Ullah Jan., Developing the Best Scheduling Algorithm from Existing Algorithms for Real-Time Operating Systems, Available at SSRN 3331997 (2019).

[10] M. Chetto., Optimal Scheduling for Real-Time Jobs in Energy Harvesting Computing Systems, in IEEE Transactions on Emerging Topics in Computing, doi: 10.1109/TETC.2013.2296537, 2(2) (2014) 122-133.

[11] Zhou, J., Real-time task scheduling and network device security for complex embedded systems based on deep learning networks. Microprocessors and Microsystems, 79 (2020) 103282.

[12] Jr E F F, Salgado R M, Ohishi T et al., Analysis of Two-Phase Flow Pattern Identification Methodologies for Embedded Systems, IEEE Latin America Transactions, 16(3) (2018) 718-727.

[13] A Mallikarjuna Reddy, Vakulabharanam Venkata Krishna, Lingamgunta Sumalatha and Avuku Obulesh., Age Classification Using Motif and Statistical Features Derived On Gradient Facial Images, Recent Advances in Computer Science and Communications, .https://doi.org/10.2174/2213275916661904171 5 1247 , 13(965) (2020)

[14] A. M. Reddy, V. V. Krishna, L. Sumalatha and S. K. Niranjan., Facial recognition based on straight angle fuzzy texture unit matrix, International Conference on Big Data Analytics And Computational Intelligence (ICBDAC), Chirala, doi: 10.1109/ICBDACI.2017.8070865, (2017) 366-372.

[15] A. M. Reddy, K. SubbaReddy and V. V. Krishna., Classification of child and adulthood using GLCM based on diagonal LBP, International Conference on Applied and Theoretical Computing

and Communication Technology (iCATccT), Davangere, doi: 10.1109/ICATCCT.2015.7457003, (2015) 857-861.

[16] Bertozzi M, Bo C, Zingaretti P. Introduction to the Special Issue on Applications of Mechatronic and Embedded Systems (MESA) in ITS, IEEE Transactions on Intelligent Transportation Systems, 19(2) (2018) 530-532.

[17] Ebert C, Dubey A., Convergence of Enterprise IT and Embedded Systems, IEEE Software, 36(3) (2019) 92-97.

[18] Huang X, Su Z, Zhang Z, et al., Vibration transmission suppression for the propeller-shaft system by hub embedded damping ring under broadband propeller forces, Nonlinear Dynamics, 91(1) (2018) 1-16.

[19] Zheng W, Wu H, Nie C., Integrating task scheduling and cache locking for multicore real-time embedded systems, Acm Sigplan Notices, 52(4) (2017) 71-80.

[20] Swarajya Lakshmi V Papineni, Snigdha Yarlagadda, Harita Akkineni, A. Mallikarjuna Reddy., Big Data Analytics Applying the Fusion Approach of Multicriteria Decision Making with Deep Learning Algorithms , International Journal of Engineering Trends and Technology, doi: 10.14445/22315381/IJETT-V69I1P204, 69(1) (2021) 24-28.

[21] Srinivasa Reddy, K., Suneela, B., Inthiyaz, S., Kumar, G.N.S., Mallikarjuna Reddy, A., Texture filtration module under stabilization via random forest optimization methodology ,International Journal of Advanced Trends in Computer Science and Engineering, doi: 10.30534/IJATCSE/2019/20832019, 8(3) (2019).

[22] A.Mallikarjuna, B. Karuna Sree., Security towards Flooding Attacks in Inter-Domain Routing Object using Ad hoc Network, International Journal of Engineering and Advanced Technology (IJEAT), 8(3) (2019).

[23] Ghobaei-Arani, M., Souri, A., Safara, F., & Norouzi, M., An efficient task scheduling approach using moth-flame optimization algorithm for cyber-physical system applications in fog computing, Transactions on Emerging Telecommunications Technologies, 31(2) (2020) e3770.

[24] Yi, N., Xu, J., Yan, L., & Huang, L., Task optimization and scheduling of distributed cyber-physical systems based on improved ant colony algorithm. Future Generation Computer Systems, 109 (2020) 134-148.

[25] Karimi, M., & Kim, H., Energy scheduling for task execution on intermittently-powered devices, ACM SIGBED Review, 17(1) (2020) 36-41.

[26] Casini, D., Biondi, A., & Buttazzo, G., Timing isolation and improved scheduling of deep neural networks for real-time systems, Software: Practice and Experience, 50(9) (2020) 1760-1777.