عدد خاص بالمؤتمر الليبي الدولي للعلوم التطبيقية والهندسية دورته الثانية (II)
2024 / 10 / 30-29

تم استلام الورقة بتاريخ: 2024/10/6م وتم نشرها على الموقع بتاريخ: 30/ 10/2024م

Optimizing Round Robin Using Machine Learning Models (KNN) in CPU scheduling

http://www.doi.org/10.62341/sana0913

Safaa Taher Abdelhameed [1], Noor Alhode Mohamed Alkikly [2], Ferdaws Omar Alkharbash[3]

[1.2.3] University of Zawia – Faculty of Science - Department of Computer Science

[1] lolasafoo172@gmail.com، [2] nooralki14@gmail.com،[3] std.100053@zu.edu.ly

Abstract

Round Robin is considered as one of the most practically recognized process scheduling algorithms in CPU scheduling because it is simple and fair. However, the efficiency of Round Robin depends a lot upon the selection of an optimal time quantum. If the quantum is too small, then frequent context switches are needed by the CPU; therefore, the overhead increases, and thus, the average waiting time for the processes also becomes long. As a result, system performance is reduced as more and more CPU time is used up in context switching, instead of task execution. it may behave similarly to the First Come First Serve (FCFS) algorithm if the time quantum is excessively long, which leads to extended average waiting time. This paper proposes an improved Round Robin algorithm by incorporating machine learning, which optimally determines the time quantum dynamically. More precisely, the K-Nearest Neighbors algorithm will be used, with NumPy in charge of data processing, for the runtime prediction of an optimal time quantum considering characteristics of processes. The experimental results showed a considerable improvement in the parameters of average waiting, turnaround time, and the number of context switches with respect to the traditional Round Robin algorithm. Results indicated that machine learning efficiently modifies the predictable scheduling algorithms to make the scheduling process adaptive and efficient in operating systems.

Key words: dynamic round robin, classical round robin, burst time, machine learning.

المجلة الدولية للعلوم والتقنية
International Science and Technology Journal
ISTJ

عدد خاص بالمؤتمر الليبي الدولي للعلوم التطبيقية والهندسية دورته الثانية (II)
2024 / 10 / 30-29

LICASE
المؤتمر الليبي الدولي للعلوم التطبيقية و الهندسية

تم استلام الورقة بتاريخ: 2024/10/6م وتم نشرها على الموقع بتاريخ: 30/ 10/2024م

تحسين خوارزمية الجولة الدائرية باستخدام نماذج التعلم الآلي (KNN) في جدولة وحدة المعالجة المركزية

صفاء الطاهر عبد الحميد [1]، نور الهدى محمد الككلي [2]، فردوس عمر الخرباش [3]

[123]جامعة الزاوية – كلية العلوم – قسم علوم الحاسوب

lolasafoo172@gmail.com، [2] nooralki14@gmail.com، [3] std.100053@zu.edu.ly

الخلاصة

يُعتبر خوارزمية الجولات الدائرية (Round Robin) واحدة من أكثر خوارزميات جدولة العمليات في جدولة وحدة المعالجة المركزية (CPU) شهرةً من الناحية العملية، لأنها بسيطة وعادلة. ومع ذلك، تعتمد كفاءة خوارزمية الجولات الدائرية بشكل كبير على اختيار فترة زمنية مثالية .(Time Quantum) إذا كانت الفترة الزمنية قصيرة جدًا، فإن وحدة المعالجة المركزية تحتاج إلى تبديل السياق بشكل متكرر؛ وبالتالي، يزداد العبء الزائد، ويصبح متوسط وقت الانتظار للعمليات طويلًا أيضًا. نتيجةً لذلك، تنخفض أداء النظام حيث يتم استهلاك المزيد والمزيد من وقت وحدة المعالجة المركزية في تبديل السياق بدلاً من تنفيذ المهام. قد تتصرف بشكل مشابه لخوارزمية من يأتي أولاً يخدم أولا (FCFS)إذا كانت الفترة الزمنية طويلة بشكل مفرط، مما يؤدي الي متوسط وقت الانتظار طويل. يقترح هذا البحث تحسينًا لخوارزمية الجولات الدائرية من خلال دمج التعلم الآلي لتحديد الوقت المثالي ديناميكيًا. بشكل أكثر تحديدًا، سيتم استخدام خوارزمية الجار الأقرب (K–Nearest Neighbors) مع مكتبة NumPy لمعالجة البيانات، للتنبؤ الفوري بوقت مثالي للفترة الزمنية بناءً على خصائص العمليات. أظهرت النتائج التجريبية تحسنًا كبيرًا في متوسط وقت الانتظار، ووقت الإنهاء، وعدد تبديلات السياق مقارنةً بخوارزمية الجولات الدائرية التقليدية. أشارت النتائج إلى أن التعلم الآلي يقوم بتعديل خوارزميات الجدولة التقليدية بكفاءة لجعل عملية الجدولة تكيفية وأكثر كفاءة في أنظمة التشغيل.

الكلمات المفتاحية: جولة روبن الديناميكية، جولة روبن الكلاسيكية، وقت التشغيل، التعلم الآلي.

Introduction

An Operating System (OS) serves as the essential interface between computer hardware and the user, managing and coordinating hardware resources among various application programs. With the evolution of modern operating systems, which have transitioned from handling single tasks to supporting multitasking environments, the role of CPU scheduling has become increasingly vital. In these environments, processes

run concurrently, and CPU scheduling determines how processes are assigned to the available CPUs, directly impacting system performance.

The primary goal of CPU scheduling is to optimize various performance metrics, such as maximizing CPU utilization and throughput while minimizing response time, waiting time, turnaround time, and the number of context switches [1]. Scheduling can be either preemptive, where a process may be interrupted to allow a higher-priority process to execute, or non-preemptive, where a process runs to completion or voluntarily vintages the CPU when it requires another resource, like input/output operations. [2].

This paper introduces a machine learning-based approach to optimize the RR algorithm. By using NumPy for data manipulation and the K-Nearest Neighbors (KNN) algorithm for predicting an optimal time quantum, we aim to dynamically adjust the scheduling parameters, improving overall system performance [3].

Among the many CPU scheduling algorithms, Round Robin (RR) is widely recognized for its simplicity and fairness. The RR algorithm assigns a fixed time quantum to each process in the ready queue, cycling through the processes in a turn-by-turn manner. However, the effectiveness of the RR algorithm is heavily dependent on the choice of time quantum. A larger time quantum can cause the system to resemble a First Come First Serve (FCFS) scheduler, while a smaller time quantum may result in excessive context switching, leading to reduced overall efficiency [5].

This paper introduces a machine learning-based approach to optimize the Round Robin algorithm by dynamically adjusting the time quantum. Utilizing NumPy for data manipulation and the K-Nearest Neighbors (KNN) algorithm for predicting an optimal time quantum, the proposed method aims to enhance overall system performance. By addressing the challenge of selecting an appropriate time quantum, this approach seeks to strike a balance between minimizing context switches and ensuring efficient CPU utilization, thereby improving the efficiency of the RR algorithm in diverse processing environments.

Preliminaries

A program is essentially a passive structure, symbolized by a file containing a series of instructions. Processes, on the other hand, are active structures that are identified by a program counter that indicates the next instruction to be performed and a collection of associated resources. An executable file becomes a process as it is put into memory, replacing its former state as a program. These processes can then be assigned different statuses and be given access to system resources, including the CPU:

NEW: A new version of the process has been developed.

RUNNING: The CPU is actively executing the process.

WAITING: The process is awaiting a happening, like the finish of an input/output operation.

READY: The process is set up and waiting on a processor.

TERMINATED: The process is no longer active after its execution is complete.

The process in the ready queue will be carried out by the processor. The process status now becomes running at this point. When the operating process is finished, it will terminate and change its status to ended. An interruption can occasionally cause the operating process to be preempted. As a result, the operating process will be forced to switch to the later-scheduled ready state. Because it is waiting for an I/O event to happen, a running process may switch to the waiting state. Once the I/O event has finished, the waiting process may return to the ready state. It should be remembered that multiple processes may be waiting and prepared at the same time, but only one process may be given a CPU at a time [6].

The term "burst time" refers to the amount of time the process needs to use the CPU. Arrival time is the moment a process enters the queue to be performed. Waiting time is the length of time a process has been waiting in the ready queue, whereas turnaround time is the total amount of time it takes to finish a certain process. Response time: It can be described as the amount of time that pass by between the process's submission of a request and its initial response. The scheduler chooses any process from queues within a routine to execute in a way that optimizes load balancing.
[1].

Related work

The Round Robin (RR) algorithm is a simple and effective CPU scheduling method that assigns a fixed time quantum to each process in a circular technique. While it is recognized for its fairness and ease of use, the RR algorithm does encounter some issues, such as increased context switching overhead when the time quantum is set too short, or it may behave similarly to the First Come First Serve (FCFS) algorithm if the time quantum is excessively long. Traditional optimization techniques for RR typically rely on static methods or situational alters. For example, [7] proposed a framework that enhances the time quantum through a grid search method. Their approach is statistically verifiable and shows better performance than existing methods. [8] presented the Variant On Round Robin (VORR-KNN) algorithm, which expands on the original RR algorithm. The VORR-KNN seeks to enhance CPU scheduling by minimizing the average waiting time, average turnaround time, and the frequency of context switches. It accomplishes this by determining the time quantum based on the median of burst times, allowing it to better adapt to different workloads.

Dynamic Round Robin (DRR) algorithms have been created to overcome the drawbacks of static time quantum. These algorithms modify the time quantum based on the remaining burst time of processes, leading to more effective scheduling. [2] introduced the "Improved Half-Life Variable Quantum Time with Mean Time Slice Round Robin CPU Scheduling (ImHLVQTRR)" algorithm, which significantly improves average waiting time and turnaround time while also reducing context switching compared to traditional RR and other existing methods. Another notable development in DRR is the "Dynamic Round Robin with Controlled Preemption" (DRRCP) algorithm, presented by [6]. DRRCP reduces unnecessary context switching by implementing a dynamic quantum time and lowers average waiting and turnaround times. Furthermore, it features

a mechanism that allows processes that have completed 95% of their execution to finish without preemption, thereby optimizing CPU resource utilization.

Advances in machine learning in recent times have created new opportunities for dynamic CPU scheduling optimization. The application of machine learning approaches to ascertain the ideal time quantum in the RR algorithm was investigated by [3]. Their study highlights how crucial time quantum is to attaining CPU use efficiency and fairness. They were able to forecast the ideal time quantum by using regression models like Polynomial and Linear Regression and classification models like Random Forest Classifier and K-Nearest Neighbors (KNN). Their findings showed that machine learning might improve system performance overall and significantly reduce average waiting times.

Even with these developments, there is still little research being done on the use of machine learning models like KNN for CPU scheduling optimization. In order to close this gap, this study presents a new approach that uses machine learning to promote more effective process scheduling by combining KNN and RR scheduling.

Methodology

We used Visual Studio Code (VS Code) for local development because of its strong Python support and sophisticated debugging features. We also utilized Google Colab, a cloud-based platform that offers free access to GPU and TPU processing capacity, to help with effective model training and experimentation.

Ten randomly generated processes are produced via a Python script we wrote to imitate the process scheduling environment. The following characteristics apply to each process:

Process ID: A special number is given to every process.

Arrival Time: Generated at random within a predetermined range to mimic when each process accesses the system.

Burst Time: A variable that is produced at random to indicate how long each process needs to execute.

The classic Round Robin (RR) scheduling method goes through each process one after the other, giving each a fixed time quantum. If a process cannot complete its execution within this time limit, it is preempted and placed to the rear of the queue. This method considers scheduling fairness, but it can lead to inefficiencies if the time quantum is not suitable for the set of operations.

Using previous process execution data, we applied a supervised learning technique, the K-Nearest Neighbors (KNN) model, to analyze and predict the ideal time quantum. With the help of this dataset, the model was trained to identify patterns and relationships within the dataset. We compared the KNN model's predictions to the results of the traditional RR algorithm under different conditions in order to validate it. By evaluating the model, it was made sure that it could adjust the time quantum in order to better suit the aspects of the process balance and increase system efficiency in general.

تم استلام الورقة بتاريخ: 2024/10/6م وتم نشرها على الموقع بتاريخ: 30/ 10/2024م

Metrics for Comparison

The performance of Classical Round Robin (CRR) and Optimized Round Robin using K-Nearest Neighbor (ORR-KNN) scheduling algorithms was evaluated using the following key metrics:

Average Waiting Time (AWT): This parameter shows how long a process typically waits in the queue before being executed. A lower AWT gives more effective process handling, making AWT an important indicator of the system's responsiveness. It is computed by taking the total number of processes and dividing it by the sum of their waiting times. Since fewer delays mean better user experience and overall system performance, a lower AWT is preferred.

Average Turnaround Time (ATT): The total duration of time needed for a process to finish, from the moment it enters the system until it is completed, is known as turnaround time. Lower values indicate faster processing. ATT gives an overall evaluation of the scheduling algorithm's efficiency. Turnaround times, or the variation between each process's arrival time and completion time, are summed up over all processes in order to calculate it. This measure is very crucial for determining how quickly the system can begin and complete tasks, especially in situations where there are a lot of processing demands.

Number of Context Switches (NCS): When the CPU transitions between processes, it saves the state of the running process and loads the state of the following one. An important metric is the overall number of context transitions made throughout the scheduling period, since too many context switches may increase burden and lower CPU efficiency. NCS was selected to assess how well CRR and ORR-KNN balance system resources with process execution, since reducing context changes is necessary for maintaining CPU performance at its most effective state.

Analysis

To evaluate how well the proposed algorithm works, let's look at some examples. We've run simulations to compare the performance of ORR-KNN with the standard RR in several scenarios taking into account when tasks arrive and how long they take to complete.

Using libraries like NumPy to manipulate data, pandas to handle data matplotlib to generate graphs and, and Scikit-learn to put the K-Nearest Neighbors (KNN) algorithm into action. We generated a random dataset with 10 processes. Each process had random arrival and burst times, which simulated a real-world workload for CPU scheduling. Fig (1) shows a Gantt chart of how CRR runs things. Fig (2) displays a Gantt chart for ORR-KNN's execution order. Table 3 compares the results of both methods.

تم استلام الورقة بتاريخ: 2024/10/6م وتم نشرها على الموقع بتاريخ: 30/ 10/2024م

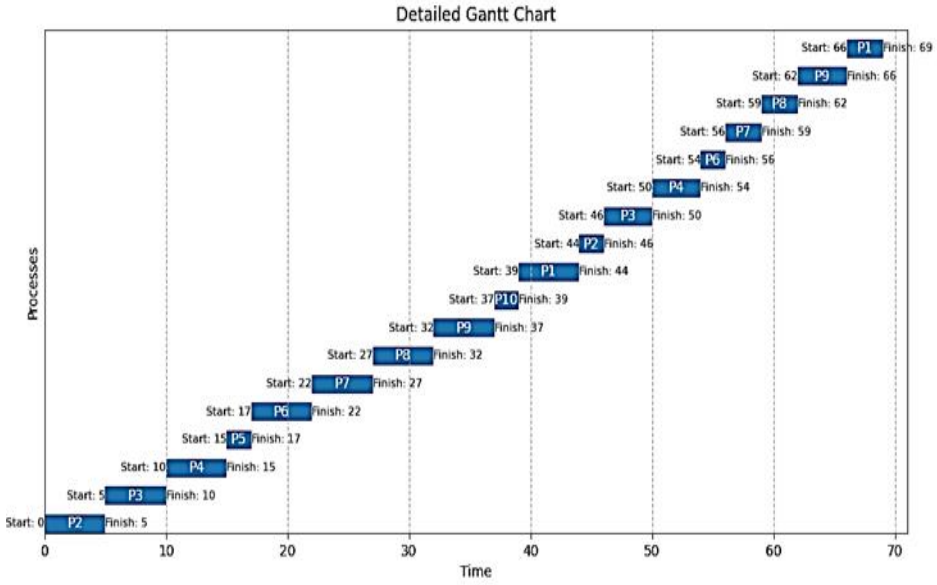

Fig.1. Gantt chart for CRR

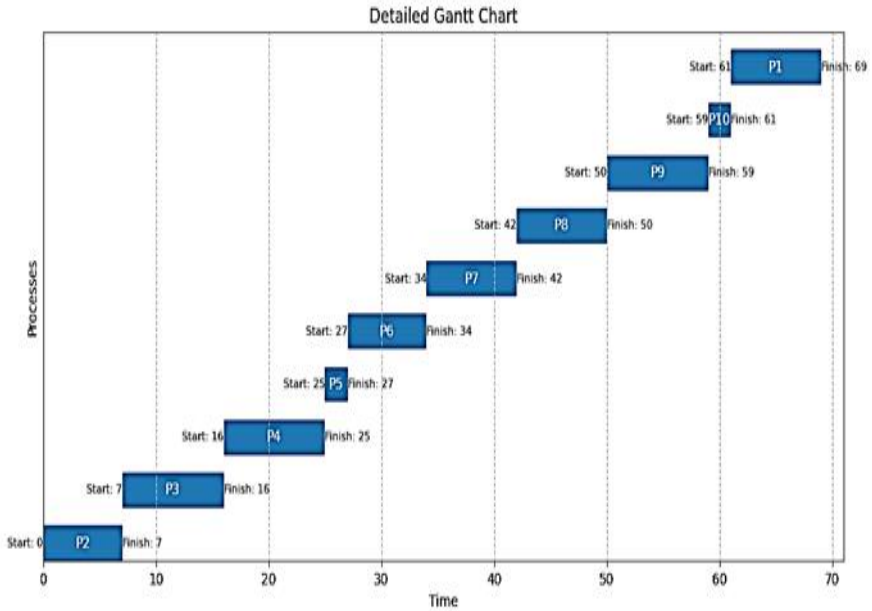

Fig. 2. Gantt chart for ORR-KNN

As shown in table 3 the comparison between CRR and ORR-KNN, the CRR shows long average waiting time, long turnaround time and more context switch. While on the other case ORR-KNN results a major reduction in average waiting time, average turnaround time and context switch.

تم استلام الورقة بتاريخ: 2024/10/6م وتم نشرها على الموقع بتاريخ: 30/ 10/2024م

Table 3 Comparison between CRR & ORR-KNN algorithm

Metric	CRR (time quantum of 5)	ORR-KNN (predicted quantum time)
Average waiting time	40.80ms	28.00ms
Average turnaround time	47.70ms	34.90ms
Number of context switch	18	10

Results and Discussion

Significant enhancements in CPU scheduling performance are shown when comparing the classic Round Robin (RR) algorithm with its improved modification, which was optimized using the K-Nearest Neighbor (KNN) machine learning model. The application of machine learning enables dynamic alterations to the time quantum, customized to the particular characteristics of the processes, leading to improved scheduling effectiveness.

Key performance parameters such as average waiting time, turnaround time, and amount of context switches all showed improvements. By analyzing previous process data and predicting an ideal time quantum for every operation, the KNN model made it possible to use CPU resources more effectively. This enhancement improves system performance overall by minimizing overhead and reducing delays.

We used these key performance indicators to compare the optimized Round Robin using K-Nearest Neighbor (ORR-KNN) algorithm versus the classical Round Robin (CRR) algorithm in order to evaluate its efficacy(see figure 3). The results of the analysis, which are displayed in Figure 4, indicated a considerable drop in the total amount of context switches as well as average waiting and turnaround times. The reliability and predictability of the mentioned advantages were confirmed by thorough tests on an array of process sets, supporting these findings.

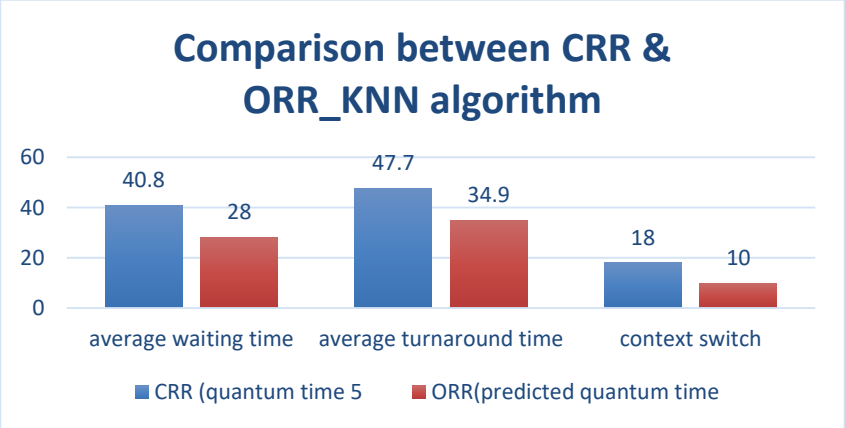

Fig .3. Comparison between CRR & ORR-KNN algorithm

تم استلام الورقة بتاريخ: 2024/10/6م وتم نشرها على الموقع بتاريخ: 30/ 10/2024م

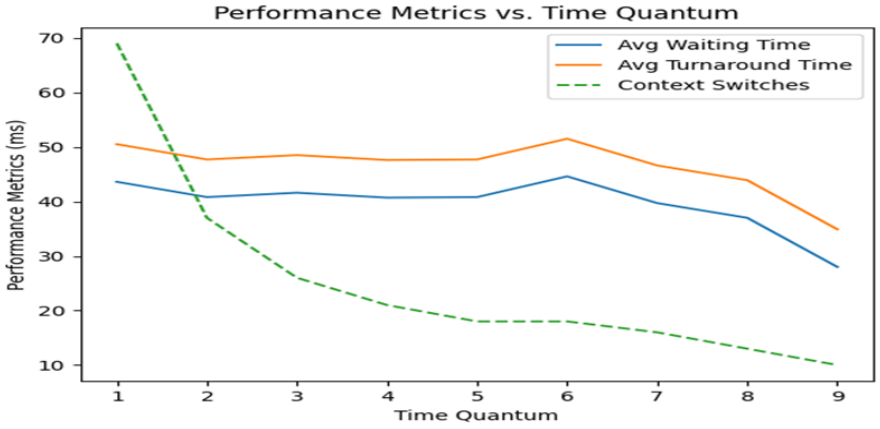

Fig .4. Performance metrics vs quantum time

The decreased level in average waiting and turnaround times indicates processes are completed faster and with fewer interruptions. Context switches are being used less frequently, which points to a more stable CPU environment with less overhead and interruptions in operations. These results highlight the value of adding KNN to the ORR algorithm and illustrate how machine learning, in comparison to standard RR techniques, can greatly improve CPU scheduling efficiency.

Conclusion

This paper presents a unique method for optimizing the round-robin scheduling algorithm with the assistance of learning machines. By participating NumPy for data manipulation and KNN for predictive modeling, dynamic data-driven optimization of time quantum becomes gathered, leading to the enhancement of the performance metrics. Our results show that machine learning is such a technique with huge potential to improve classic scheduling algorithms. Some generalizations that might be done in the future work are in relation to other machine learning models or an improvement in the optimization process.

References

[1]. H. Behera, R. Mohanty, S. Sahu, and S. Bhoi, "Comparative performance analysis of multi dynamic time quantum Round Robin (MDTQRR) algorithm with arrival time," *Indian Journal of Computer Science and Engineering*, vol. 2, Sep. 2011.

[2]. A. Simon, G. Dams, and S. Danjuma, "An improved half-life variable quantum time with mean time slice round robin CPU scheduling (IMHLVQTRR)," *Science World Journal*, vol. 17, p. 2022, Jul. 2022.

[3]. P. Chitaliya, S. Kulkarni, A. Telang, D. Zaveri, A. Shah, and K. Deulkar, *Time Quantum Optimization in Round Robin Algorithm*. 2023,

[4]. A. Noon, A. Kalakech, and S. Kadry, "A new Round Robin-based scheduling algorithm for operating systems: Dynamic quantum using the mean average," *International Journal of Computer Science Issues*, vol. 8, Nov. 2011.

تم استلام الورقة بتاريخ: 2024/10/6م وتم نشرها على الموقع بتاريخ: 30/ 10/2024م

[5]. C. S. Yusuf and S. Abdullahi, A Grouped Half-Life Variable Quantum Time Round Robin Scheduling (GHLVQTRR) Algorithm for CPU Process. 2016.
[6]. A. Simon, A. Saleh, and S. Junaidu, "Dynamic Round Robin with Controlled Preemption (DRRCP)," International Journal of Computer Science Issues, vol. 11, pp. 109–117, Nov. 2019.
[7]. A. Gupta, P. Mathur, C. Travieso, M. Garg, and D. Goyal, *ORR-KNN: Optimized Round Robin CPU Scheduling Algorithm*, pp. 296–304, Aug. 2021, Doi: 10.1145/3484824.3484917.
[8]. A. Abdelhafiz and A. Afaf, "VORR-KNN: A New Round Robin Scheduling Algorithm," Al-Azhar Bulletin of Science, vol. 32, no. 2, Art. no. 12, 2021. doi:10.21608/absb.2021.99340.1141.