Reviewing And Analysis of The Deadlock Handling Methods

Enas E. El-Sharawy[1*] , Thowiba E Ahmed[2], Reem H Alshammari[3] ,
eeelsharawy@iau.edu.sa ,teahmed@iau.edu.sa, ralshammari@iau.edu.sa
Wafaa Alsubaie [4] ,Norah Alqahtani [5] , Asma Alqahtani [6]
2180006511@iau.edu.sa, 2180001045@iau.edu.sa; 2180001191@iau.edu.sa
[1,2,3,4,5,6] , Computer Science Department, College of Science and Humanities, Imam Abdulrahman Bin Faisal University, P.O.Box 31961, Jubail, Saudi Arabia

Abstract
Objectives: The primary goal of this article is to compare the multiple algorithms used for deadlock handling methods and then outline the common method in deadlock handling methods. Methods: The article methodology begins with introducing a literature review studying different algorithms used in deadlock detection and many algorithms for deadlocks prevented, recovered, and avoided. Discussion and analysis of the literature review were done to classify and compare the studied algorithms. Findings: The results showed that the deadlock detection method solves the deadlock. As soon as the real-time deadlock detection algorithm is identified and indicated, it performs better than the non-real-time deadlock detection algorithm. Our novelty the statistics that we get from the percentages of reviewing outcomes that show the most effective rate of 47% is in deadlock prevention. Then deadlock detection and recovery with 28% finally, a rate of 25% for deadlock avoidance.

Keywords:
Operating System, CPU Management, Deadlock Handling Methods, Deadlock Prevention, Deadlock Avoidance, Deadlock Detection, And Recovery.

1. Introduction

Operating systems are like an interface between the user and the computers. Without it, humans cannot manage, control, and use computer systems and assist in allocating resources such as memory, disk, files, and many of them, which help solve the user's problems and make the system usable effectively. To the CPU (the Central Processing Unit), management is essential to achieve maximum CPU usage by scheduling the CPU, allowing one process to use the CPU. At the same time, there is a process waiting due to a lack of resources [1]. Deadlock is one of the expected severe problems in various operating systems and multi-tasking programs [2]. Deadlock occurs when a process is waiting. At the same time, another process holds a requested resource, which awaits a resource from another process. And therefore, none of the processes can run and release retained resources in an operating system. So, if the process cannot change its state indefinitely due to the use of required resources by another process, the system becomes deadlocked. Deadlocks are an unwelcome yet intriguing condition of the system. Several variable conditions can lead to deadlocks, which, as a result, make them highly complicated and challenging to stop. Recovering from deadlocks often has its difficulties, mainly since it is difficult to re-establish the mechanism, complete broken processes, and a comparable one before the deadlock. Through resource trajectories and state diagrams, several formalized approaches can forecast deadlocks. However, it needs a strong.

There are three common strategies for dealing with a deadlock: deadlock prevention, deadlock avoidance, and deadlock recovery [2]. The built operating system and highly effective synchronization algorithms limit their occurrence [3] entirely. In this research paper, we have surveyed deadlock. This paper is structured as follows (I) Literature Selection Methodology, (II) Related Work, (III) Discussion, and (IV) Conclusion.

2. literature selection methodology

Literature selection methodology is the methodology that was followed in literature selection as follows:

A-The keyword search phase: The research articles were searched in the Google Scholar database using the following keywords: (1) Operating Systems (2) CPU management (3) Deadlock Methods (4) Deadlock avoidance (4) Deadlock recovery (5) Deadlock prevention. The result of the research at this stage is 55 articles.

B- Abstract Filter phase: The abstract determines the research articles relevant to the research at this stage. The selection resulted in 32 articles on deadlock avoidance, deadlock recovery, and deadlock prevention.

2.1. Related work

In this section, the deadlock handling methods are reviewed as follows:

2.1.1 Deadlock prevention

Manuscript received October 5, 2022
Manuscript revised October 20, 2022
https://doi.org/10.22937/IJCSNS.2022.22.10.30

In [2], the researchers have considered deadlock one of the most severe database problems, and if the platform is distributed, the matter becomes more complex. The paper's main goal was to improve other deadlock prevention algorithms and propose the VGS algorithm, which prevents deadlocks for system applications distributed by a dual-commitment protocol. The transactions were carried out through the pipeline method, effectively preventing deadlocks. The mechanism reduces wait time for required transactions and enables transactions to share the same set of resources. The proposed algorithm eliminates the circular wait state for the shared resource pool operations. Two commitment protocols prevent wait and wait for states without causing any low resource usage.

In [4], the researchers have considered the deadlock problem more complex if the platform is distributed. Therefore, it is essential to develop an effective control system to improve system performance while preventing deadlock. The paper's main objective is an algorithm proposal to prevent deadlock end with the advent of grid computing, prevent the problem of deadlock in the network environment by resource availability, and ensure that all available services in the network are used efficiently. The technology applied is C-platform encryption; the AMD A6-3420M APU is the processor used. The RAM is 4 GB, and the operating system is Windows 7. The system provides an improvement over other deadlock prevention algorithms [2], while [4] using an algorithm had proposed to prevent a dead end with the advent of grid computing. The results indicate that the program code did run in the university lab and on personal systems. It did implement using a particular network simulation. The researcher recommends that many experiments should be carried out to verify the results obtained.

In [5], effective priority-based and voting algorithms for deadlock prevention in a distributed environment are introduced to solve distributed systems' concurrency and starvation control problems. The authors in [7] provide deadlock prevention mechanisms of grid computing, while this study presents a solution for distributed systems against the starvation and concurrency control problems. Besides mentioning the characteristics of the priority-based and voting algorithms and some novel features like implementing non-maskable SJSF and interrupts techniques for sharpening the search to pick the most outstanding candidate for accessing the critical section (CS). The introduced algorithms afford a better opportunity for deadlock prevention by allowing one process in every CS. It presents a fair chance to all the participating processes for accessing CS.

In [6], the researchers have considered that shifting the cloud computing model leads to disruptions to traditional business models in the information and communication sector. So, this paper aims to establish a standstill prevention algorithm (n VM-out-1 PM) for resource separation and virtual machine management. The algorithm works on heterogeneous distributed platforms to reschedule sourcing policies for resource specifications. The anti-freeze algorithm has been implemented on a physical machine server with an Intel E5-2603V3 processor and 16GB of memory. During the day, the RAM, CPU, and complex drive data have been collected from the device's physical servers every 4 hours. The proposed algorithm was implemented using the Cloud Sim emulator. The results showed that the anti-freeze algorithm has the best performance and the most beneficial material resources. In [4] and [6], they are similar in creating a deadlock algorithm but in [4] in preventing deadlock with grid computing, while [6] for resource separation and management in virtual machines, on heterogeneous distributed platforms.

In [7], the authors have considered the Deadlock problem. Multiple transactions can access one database simultaneously through a concurrent transaction database system. This allows different transactions to join the same data object simultaneously. The author suggested the Petri net model of the 2PL locking protocol for concurrent transaction databases to solve this problem. Next, give the 2PL protocol's anti-deadlock algorithm. On this basis, a shatter-resistant scheduling system provides using this protocol. This technology is particularly suitable for preventing deadlock in distributed concurrent transaction databases. State equations are used to evaluate a correct scheduling method to avoid stalemates and ensure the smooth execution of simultaneous transactions and access graphs. The main conclusion of this study is to prevent stalemates, and this study addresses deadlocks. Compared to [6], the device deals with inertia in cloud computing.

In [8], the deadlock problem can arise when multiple processes try to access shared resources if some basic synchronization mechanisms are adopted. Therefore, the authors proposed a new solution to distributed transaction services and several types of deadlocks. They have used advanced replica and timestamp mechanisms to prevent deadlocks. This approach uses existing lock-in policies with replicas to solve foolproof deadlocks. The experimental results showed that there is no dead end. A timestamp-based restart policy is more convenient and outperforms similar solutions to distribute resources globally. The proposed mechanism does not include more precise control over many locks, such as reciprocal and proprietary locks. Unlike [7], the system deals with several types of locks. The resource manager must handle standard locks with particular care, but they require more careful consideration.

In [9], the authors have addressed deadlock control of multithreaded software Dependence on Gadara nets. The method they used is an iterative deadlock prevention policy based on siphons. They have successfully figured out all the false markings that reduce emptiable siphon and are forbidden to avoid deadlocks.

In [10], the authors tackled the deadlocks problem in the flexible manufacturing system (FMS). They have used an improved deadlock prevention strategy that depends on controlling to avoid deadlocks. The result has avoided all false markings; hence, the deadlock issue improved. In contrast with [9], the method was less complex due to the enhancement as they figured out that siphons with more resources can be controlled by managing those with fewer resources.

In [11], the authors had a problem with the increasing need for dynamic transmission parameters in modern wireless communication standards. That caused the efficiency of the traditional static scheduling method to reduce by applying Odyn technology which proposes a mechanism to prevent the deadlock resulting from attempts to allocate buffers in limited memory. Using the Design Space Exploration design tool, the results show that it can combat deadlock in offline mode.

In [12], the authors have presented an innovative design method based on PN to prevent FMS (Flexible Manufacturing System) inertia's Deadlock by integrating and implementing one-step forward and one-step backward-looking algorithms and using the simplified constant-based method. This method led to more permissive behavior of the controlled closed-loop system. Both the [9] and [11] deadlock algorithms; however, the first one uses a combination of one-step forward and one-step backward algorithms. While the second implements Odyn technology, which proposes a mechanism to prevent deadlocks resulting from attempts to allocate buffers in limited memory.

In [13], the authors have considered the problem of deadlocks in production software. The authors' primary technique for deadlock detection and prevention in production software is the UNDEAD system. It also proposes a new idea of "merging locks" to tentatively prevent deadlock problems, providing a "band-aid" for production software. The evaluation results show that the UNDEAD system detects all known stalemates. In the study [6], a new algorithm for deadlock prevention is. While [13], the detection and prevention of deadlocks.

In [14], the authors have viewed a potential dilemma and the lack of means for modifying information between separate applications due to the complexity of a typical allocation of network resources covering multiple administrative areas. Therefore, they proposed a new approach that helps in the rapid joint allocation of network resources: this approach is based on freedom from inertia and building on the atomic deal. A test did perform to evaluate the approach's effectiveness by simulating a Common Network Resource Simulation System (GRCS), running simulations 20 times for each case. The results showed that the proposed method outperformed ODP2 and improved significantly. We have seen that the atomic treatment approach is efficient and has demonstrated this in simulated experiments. As in [8], the method used is excellent and has shown impressive results in preventing a dead end in a different area.

In [15], the authors have considered the deadlocks that can occur in an AMS during its operation by shared resources rivalry, which can cause disastrous results in automated manufacturing systems. This study aims to evaluate the performance of automated manufacturing systems by using Petri nets under different deadlock prevention policies. The primary technique for deadlock prevention is Siphons and iterative methods.

They used a simulation tool for 24-time units; the result shows that the strict minimal siphon method in an extensive system leads to the best behavioral permissiveness than the other methods. In [14], the simulation method and the suggested approach achieved satisfactory results, while in [15], the Siphons method was better than the other methods.

In [16], the authors have been studying the problem of deadlock in the data center network. The study aims to Debate the issue of deadlock formation in lossless networks. In this study, the problems of the deadlock that occur are discussed, and their causes are. The study suggests some methods that help prevent deadlock before it happens, using the simulation method. The result shows they are many reasons for deadlock .in [15] was used different deadlock prevention policies were, but [16] discussed the problems of deadlock that occur and their causes.

In [17], the authors proposed a new method for preventing deadlocks in synchronous systems, a group of processes that conservatively share a set of common resources. It can be implemented by solving a set of linear integer programming problems. The results showed that this method applies to more complex systems. This study's problem is the same as the one in the study [8], but the process of solving it differs as they used advanced replicas and timestamp mechanisms to prevent stalemates.

2.1.2 Deadlock avoidance

In [18], the authors have extended the classic Banker's algorithm to a class of flexible manufacturing systems modeled by Petri Networks. It possesses two properties: the first is to allow flexible routing of parts, and the second is to allow multiple sets of resources to be used in each processing step to avoid inertia. And to optimize Banker's algorithm approach rather than guaranteeing resources Sufficient to finish processing a specific part, it is sufficient to transfer the part to a state in which there is no interaction with the rest of the parts. This study differs in that it used the classic Banker's algorithm from the dynamic Banker's algorithm used in the study [21], but both algorithms perform the same purpose: to avoid deadlock.

In [19], the authors have considered the problem of unreliable communications resulting from the deadlocks of industrial IoT devices that use multiple computing Access (MEC) for bankers. So, they proposed the proposed resource-saving algorithm to avoid deadlock; this algorithm is based on fetching jobs from the RMS Scheduled Tasks queue. This algorithm proceeds that the most used tasks get a higher priority. Then it moves the jobs from the job queue to the ready queue. The algorithm modifies how the tasks leave the job queue and remain ready-made. The simulation of this algorithm was made and compared in a system with and without using the RPA algorithm. The results helped prevent deadlock and provide more reliable interaction in the network between mobile stations and MEC platforms and based on Optimizing system power by getting rid of inertia. From our saw effort and work, they thank it and produced beneficial results in multi-access computing.

In [20], the authors have considered the problem of poor resource management in cloud computing that leads to Dead Lock. So, they have proposed an algorithm that helps in load balancing, which is the throttling algorithm based on preventing exceeding the task queue to balance the load and avoid disasters. Also, they implemented it by simulating with the help of the Cloud tool Analyst, which provides me with a visualization of the results after comparing the system with and without load balancing. The results showed that load balancing is the most effective technique for preventing disasters. So, the algorithm maintains the state of resources periodically, and the computing model has improved productivity response time to operations.

In [21], the researchers have considered Banker's algorithm to allocate resources and avoid pitfalls. The study's main aim is to suggest an approach to Banker's algorithm to solve a problem related to determining the cause of failure in performing operations. The proposed Dynamic Banker algorithm resolves this problem after comparing the performance of both algorithms. The results showed that the proposed algorithm avoids stalemate by identifying the process that needs an increase in resources and the ease of adding resources. A throttling algorithm has been submitted to aid load balancing based on task overrun prevention [20]. Whereas [21] presented an improved approach to Banker's algorithm.

In [22], the authors have considered the problem of ineffective resource allocation that causes the problem of complete stoppage of operations within the system. Given the complexity of the algorithms that help in avoiding these problems, such as the banker algorithm, they proposed a new method that is less complex than the banker algorithm and other algorithms. This method relies on the ease of verifying resource requests. The new algorithm has compared it to the Banker algorithm's performance. The result shows that the proposed method is lower than the Banker algorithm's complexity. The proposed method has surpassed the suggested method in [21] effectiveness, time, and complexity.

In [23], the authors have considered the Deadlock issue that arises in modem flexible manufacturing systems (FMS) easily because of shared source usage and high production flexibility. This study aims to develop a novel deadlock avoidance algorithm, and the primary technique is by calculating the available usable space of circuits of the digraph model. The result shows the algorithm secures high permissiveness based on the dynamic, practical free space calculation of circuits in the digraph model based on the average percentage, which allows it to be above 90% among all the tested examples consistently. In [21] propose an approach to the Banker algorithm**.**

In [24], the authors discuss the problem of bankers' security algorithm whether the system is in a safe or insecure state, based on the needs of operations and their availability using the permutation tree and the solution tree. The results showed that the solution tree would have fewer states than the permutation tree. As in [21], it's using the same algorithm, but with a different methodology that determines the cause of failure in operations; its results showed that the algorithm Can avoid the impasse by the process that needs an increase in resources and the ease of adding resources to it.

In [25], the authors have considered the problem of inertia in an automated guided vehicle (AGV) system. A proposal that avoids rigidity is an adaptable and straightforward algorithm based on a theoretical graph approach instead of complex and immutable methods like Petri-net. The work here has been carried out in two phases: the first phase deals with avoiding the deadlock algorithm, and the second part proposes targeting strategies that fit the proposed algorithm. This algorithm has been characterized by the ease of modifying the model and configuring the system. The mathematical tests have been performed on 100 different seeds, and the results yielded a high performance in avoiding inertia. And we see that their efforts deserve thanks in applying a method to prevent stagnation less than Petri-net's complexity as in [7].

2.1.3 Deadlock detection and recovery

In [26], the authors have considered the state space explosion while using modelmodel-checkingniques. The model checker consumes all the available memory to produce reachable states. The main methods applied proposed by the authors for deadlock detection in complex software systems they a hybrid algorithm that uses Particle swarm optimization (PSO) and BAT (BA) (BAPSO), and a greedy algorithm to detect deadlocks. The empirical findings indicate that the hybrid algorithm (BAPSO) is more efficient than the particle swarm optimization algorithm (PSO) and BAT algorithm (BA). And regarding the greedy algorithm can compete with the meta-heuristic algorithms in terms of accuracy and speed.

Deadlock detection algorithms have been proposed in complex software systems [26], while [27] deadlock detection algorithms in large-scale operating systems. In [27], the authors have considered the deadlocks of operating systems. So, they proposed the Race Condition Based Deadlock Detection methodology by using queues to detect race conditions causing deadlocks, which can appear in the subsequent execution. As a result, the algorithm has proven helpful since it is scalable when applied in a large-scale environment. Furthermore, it is compatible with parallel applications working as a master/slave mechanism. In contrast, [26] was explicitly determined to study the need to resolve deadlocks in Graph Transformation Systems (GTS). This research was for a broader scope: large-scale operating systems.

In [28], the researchers have considered that resource allocation at the infrastructure level is essential in assigning physical resources to virtual resources to more helpful use resources in the cloud computing environment. This paper's main objective is to suggest a new algorithm for allocating resources to infrastructure based on a deadlock detection approach. They used an open-source platform heterogeneous and Java language to program algorithm implementation. The proposed algorithm has been implemented using the Cloud Sim simulator. The results showed that the proposed algorithm could quickly detect stalemates and solve practical situations. In contrast,[27] deadlock detection algorithms have been implemented in large-scale operating systems. While [28] has been proposed for the deadlock detection algorithm.

In [29], the authors have considered the deadlocks problem concerning resource allocation in heterogeneous platforms. The primary applied technique proposed to solve the deadlock problem by the authors is an approach for developing the parallel deadlock detection algorithm. The results show that applying suitable scheduling algorithms for distributed resources of virtual server systems provides optimal execution. The study [28] used a new algorithm that has been proposed to detect deadlock. While [29] advancing the deadlock detection algorithm.

In [30], the authors have considered the deadlock problem in distributed systems; a distributed deadlock detection algorithm based on history-based edge chasing has been proposed. The algorithm can handle the simultaneous implementation of the algorithm and evade the same deadlock state's detection. The suggested algorithm resolved and detected the deadlock in Distributed Systems, unlike [28], which detects the deadlock in the cloud computing environment.

In [31], the authors have discussed the deadlock recovery strategy with minimal buffers for unified automatic material handling systems (UAMHSs). Recent technologies have proposed a Deadlock detection model for UAMHSs. The appropriate criteria for device deadlocks based on actual UAMHS properties have been established along with a new deadlock recovery technique. Besides, for parallel solving UAMHS deadlocks, an efficient heuristic algorithm is suggested. Simulation trial findings indicate that the novel deadlock recovery approach is superior to the benchmark approach in reducing deadlock time and enhancing tools. The study [31] detected and recovered the deadlock while it [26][27] [29] only detected the deadlock.

In [32], the authors have considered one of the leading design problems in the real-time concurrency control of transactions: local and distributed resolution while meeting the timing requirements of transactions. Therefore, a new deadlock detection algorithm has been proposed, specially built for distributed real-time database systems. They carried out extensive simulation experiments to evaluate the performance of the proposed algorithm. The findings show that the real-time deadlock detection algorithm works much better than the algorithm for non-real-time deadlock detection. Unlike [28], a study focuses on detecting deadlock in the cloud computing environment. This study focuses on the management of real-time transaction concurrency.

In [33], the authors have considered distributed deadlock problem. A new algorithm for distributed deadlock detection has been suggested. The Deadlock was detected along the edges of the wait-for graphs in the scheme by forwarding messages, called probes. The algorithm is an updated variant of the algorithm of Chandy that, in some circumstances, is false. The performance findings suggest that the probe initiation rate is a dominant factor in evaluating system performance in the modified algorithm. The probe-based algorithm will outperform the time-out technique for significant multiprogramming level values compared with the time-out technique.

In [34], the authors have discussed and analyzed the issue of deadlocks in distributed DBMSs. The two predominant deadlock models and the four different distributed approaches to deadlock detection are discussed in these systems. The Suggested methodology in this study is a new deadlock detection algorithm. The algorithm was based on the dynamic creation of deadlock detection agents (DDAs), each responsible for detecting deadlocks in the global wait-for-graph component of one connected component (WFG). The experimental results indicate that our newly proposed deadlock detection algorithm outperforms the other algorithms in most configurations and workloads and is very robust concerning different load and access profiles, in contrast to all other algorithms.

3. Results and Discussion

After reviewing the studies for the deadlock handling methods**, many** techniques have been developed to solve deadlock problems. The reviewed studies from [2] to [17] based on deadlock prevention utilized timestamps and advanced replicas techniques to prevent different deadlock types. And the other presented a Petri net model of the locking protocol 2PL concerning a concurrent transaction database to solve the deadlock problem. The findings showed that the deadlock prevention system minimized and avoided deadlock after using the suggested prevention mechanisms.

The reviewed studies from [18] to [25], based on deadlock avoidance, handled the deadlock using resource-saving to avoid deadlock. Other studies attempted to improve Banker's security algorithm for allocating resources and avoiding deadlocks. The results showed high performance in avoiding the impasse by identifying the process that needs increased resources.

The reviewed studies from [26] to [34] based on deadlock detection and recovery used the queues to detect the deadlocks appearing in the subsequent execution. And some others suggest an algorithm for allocating resources to infrastructure based on a deadlock detection approach. These paper presented studies related to the deadlock problem to review deadlock algorithms to handle the deadlock problem more efficiently. The results showed that the deadlock detection method solves the deadlock. As soon as the real-time deadlock detection algorithm is identified and indicated, it performs better than the non-real-time deadlock detection algorithm. The percentages of reviewing outcomes are shown. The most significant percentage of 47% is reached in deadlock prevention. Then deadlock detection and recovery with 28%—finally, a percentage of 25% for deadlock avoidance.

4. Conclusion

The results showed that the deadlock detection method solves the deadlock. Our findings are the statistics that show the largest percentage is reached in deadlock prevention. Then deadlock detection and recovery with 28% finally, a percentage of 25% for deadlock avoidance. This paper presented studies related to multiple algorithms used for deadlock handling methods: detection and recovery, avoidance, and prevention. After discussion and analysis of the literature review have been done to classify and compare the algorithms studied, we found that the deadlock detection method solves the deadlock. As soon as the real-time deadlock detection algorithm is identified and indicated, it performs better than the non-real-time deadlock detection algorithm. The percentages of reviewing outcomes are shown. The largest percentage of 47% is reached in deadlock prevention. Then deadlock detection and recovery with 28% finally, a percentage of 25% for deadlock avoidance. We can increase the reviewed paper and compare another comparison aspect for future work.

References

[1] Silberschatz A, Galvin PB, Gagne G. Operating system concepts. John Wiley & Sons; 2006 Jul 13. https://go.exlibris.link/0l5YRLFy.

[2] Goswami V, Singh A. VGS algorithm: an efficient deadlock prevention mechanism for distributed transactions using pipeline method. International Journal of Computer Applications. 2012 May;46(22):1-9.doi:10.5120/7094-9224.

[3] Dimitoglou G. Deadlocks and methods for their detection, prevention, and recovery in modern operating systems. Operating systems review. 1998 Jul 1;32(3):51-4.doi:10.1145/281258.281273.

[4] Malhotra D. Deadlock prevention algorithm in a grid environment. In MATEC Web of Conferences 2016 (Vol. 57, p. 02013). EDP Sciences. DOI:10.1051/matecconf/20165702013.

[5] Mishra KN. Efficient voting and priority-based mechanism for deadlock prevention in distributed systems. In2016 International Conference on Control, Computing, Communication, and Materials (ICCCCM) 2016 Oct 21 (pp. 1-6). IEEE. DOI: 10.1109/ICCCCM.2016.7918267

[6] Nguyen HH, Nguyen TT. Deadlock prevention for resource allocation in model nVM-out-of-1 PM. In2016 3rd National Foundation for Science and Technology Development Conference on Information and Computer Science (NICS) 2016 Sep 14 (pp. 246-251). IEEE. DOI:10.5120/7094-9224.

[7] Xiao Ling Y. A Deadlock Prevention Algorithm for The Two-Phase Locking Protocol Based on Petri Net. In2019 6th International Conference on Systems and Informatics (ICSAI) 2019 Nov 2 (pp. 889-892). IEEE. DOI: 10.1109/ICSAI48974.2019.9010538.

[8] Lou L, Tang F, YouI, Guo M, Shen Y, Li L. An Effective Deadlock Prevention Mechanism for Distributed Transaction Management. In2011 Fifth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing 2011 Jun 30 (pp. 120-127). IEEE. DOI: 10.1109/IMIS.2011.109.

[9] Duo W, Jiang X, Karoui O, Guo X, You D, Wang S, Ruan Y. A deadlock prevention policy for a class of multithreaded software. IEEE Access. 2020 Jan 6;8:16676-88. DOI: 10.1109/ACCESS.2020.2964312.

[10] Zhuang Q, Dai W, Wang S, Du J, Tian Q. A MIP-based deadlock prevention policy for siphon control. IEEE Access. 2019 Sep 6;7:153782-90.doi: 10.1109/ACCESS.2019.2939855.

[11] Dauphin B, Pacalet R, Enrico A, Apvrille L. Odyn: Deadlock Prevention and Hybrid Scheduling Algorithm for Real-Time Dataflow Applications. In2019 22nd Euromicro Conference on Digital System Design (DSD) 2019 Aug 28 (pp. 88-95). IEEE. DOI: 10.1109/DSD.2019.00023.

[12] Zeng G, Wu W, Zhou M, Mao W, Su H, Chu J. Design of Petri net-based deadlock prevention controllers for flexible manufacturing systems. In2009 IEEE International Conference on Systems, Man and Cybernetics 2009 Oct 11 (pp. 193-198). IEEE. DOI: 10.1109/ICSMC.2009.5346582.

[13] Zhou J, Silvestro S, Liu H, Cai Y, Liu T. Undead: Detecting and preventing deadlocks in production software. In2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE) 2017 Oct 1 (pp. 729-740). IEEE. DOI: 10.1109/ASE.2017.8115684.

[14] Chuanfu Z, Yunsheng L, Tong Z, Yabing Z, Kedi H. A Deadlock Prevention Approach based on Atomic Transaction for Resource Co-allocation. In2005 First International Conference on Semantics, Knowledge, and Grid 2005 Nov 27 (pp. 37-37). IEEE. DOI: 10.1109/SKG.2005.4.

[15] Abou el Nasr E, El-Tamimi AM, Al-Ahmari A, Kaid H. Comparison and evaluation of deadlock prevention methods for different size automated manufacturing systems. Mathematical Problems in Engineering. 2015 Sep 16;2015. https://doi.org/10.1155/2015/537893

[16] Hu S, Zhu Y, Cheng P, Guo C, Tan K, Padhye J, Chen K. Deadlocks in data center networks: Why do they form, and how to avoid them. InProceedings of the 15th ACM Workshop on Hot Topics in Networks 2016 Nov 9 (pp. 92-98). https://doi.org/10.1145/3005745.3005760.

[17] Tricas F, Garcia-Valles F, Colom JM, Ezpeleta J. A Petri net structure-based deadlock prevention solution for sequential resource allocation systems. InProceedings of the 2005 IEEE international conference on robotics and automation 2005 Apr 18 (pp. 271-277). IEEE. doi:10.1109/ROBOT.2005.1570131.

[18] Ezpeleta J, Tricas F, Garcia-Valles F, Colom JM. A banker's solution for deadlock avoidance in FMS with flexible routing and multi-resource states. IEEE Transactions on Robotics and Automation. 2002 Dec 10;18(4):621-5. DOI: 10.1109/TRA.2002.801048.

[19] Ugwuanyi EE, Ghosh S, Iqbal M, Dagiuklas T. Reliable resource provisioning using bankers' deadlock avoidance algorithm in MEC for industrial IoT. IEEE Access. 2018 Aug 10;6:43327-35.doi: 10.1109/ACCESS.2018.2857726.

[20] Mahitha O, Suma V. Deadlock avoidance through efficient load balancing to control disaster in a cloud environment. In2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT) 2013 Jul 4 (pp. 1-6). IEEE. doi:10.1109/ICCCNT.2013.6726823.

[21] Gaur M, Singh D. Implementation of Banker's Algorithm Using Dynamic Modified Approach. International Journal on Recent and Innovation Trends in Computing and Communication.;5(11):157-63. https://d1wqtxts1xzle7.cloudfront.net.

[23] Begum M, Faruque O, Miah MW, Das BC. An Improved Safety Detection Algorithm Towards Deadlock Avoidance. In2020 IEEE 10th Symposium on Computer Applications & Industrial Electronics (ISCAIE) 2020 Apr 18 (pp. 73-78). IEEE. DOI: 10.1109/ISCAIE47305.2020.9108818.

[24] Zhang W, Judd RP. Deadlock avoidance algorithm for flexible manufacturing systems by calculating effective free space of circuits. International Journal of Production Research. 2008 Jul 1;46(13):3441-57.DOI: 10.1109/ACC.2005.1470589.

[25] Singh RR, Singh DK. Deadlock Avoidance: A Dynamic Programming Approach. In2010 International Conference on Computational Intelligence and Communication Networks 2010 Nov 26 (pp. 661-664). IEEE. DOI: 10.1109/CICN.2010.130.

[26] Yoo JW, Sim ES, Cao C, Park JW. An algorithm for deadlock avoidance in an AGV System. The International Journal of Advanced Manufacturing Technology. 2005 Sep;26(5):659-68. https://doi.org/10.1007/s00170-003-2020-4

[27] Yousefian R, Aboutorabi S, Rafe V. A greedy algorithm versus metaheuristic solutions to deadlock detection in Graph Transformation Systems. Journal of Intelligent & Fuzzy Systems. 2016 Jan 1;31(1):137-49. https://doi.org/10.3233/IFS-162127

[28] Do-Mai AT, Diep TD, Thoai N. Race condition and deadlock detection for large-scale applications. In2016 15th International Symposium on Parallel and Distributed Computing (ISPDC) 2016 Jul 8 (pp. 319-326). IEEE. DOI: 10.1109/ISPDC.2016.53.

[29] Nguyen HH, Dang HV, Pham NM, Le VS, Nguyen TT. Deadlock detection for resource allocation in heterogeneous distributed platforms. recent Advances in Information and Communication Technology 2015 2015 (pp. 285-295). Springer, Cham. https://doi.org/10.1007/978-3-319-19024-2_29

[30] Nguyen HH, Nguyen TT. The algorithmic approach to deadlock detection for resource allocation in heterogeneous platforms. In2014 International Conference on Smart Computing 2014 Nov 3 (pp. 97-103). IEEE. DOI: 10.1109/SMARTCOMP.2014.7043845.

[31] Farajzadeh N, Hashemzadeh M, Mousakhani M, Haghighat AT. An efficient generalized deadlock detection and resolution algorithm in distributed systems. InThe Fifth International Conference on Computer and Information Technology (CIT'05) 2005 Sep 21 (pp. 303-309). IEEE. DOI:10.1109/CIT.2005.69.

[32] Zhou Q, Zhou BH. A deadlock recovery strategy for unified automated material handling systems in 300 mm wafer fabrications. Computers in Industry. 2016 Jan 1;75:1-2. https://doi.org/10.1016/j.compind.2015.10.014.

[33] Yeung CF, Hung SL. A new deadlock detection algorithm for distributed real-time database systems. InProceedings. 14th Symposium on Reliable Distributed Systems 1995 Sep 13 (pp. 146-153). IEEE. DOI: 10.1109/RELDIS.1995.526222.

[34] Obermarck R. Distributed deadlock detection algorithm. ACM Transactions on Database Systems (TODS). 1982 Jun 1;7(2):187-208. https://doi.org/10.1145/319702.319717.

[35] Krivokapić N, Kemper A, Gudes E. Deadlock detection in distributed database systems: a new algorithm and a comparative performance analysis. The VLDB Journal. 1999 Oct;8(2):79-100. https://doi.org/10.1007/s007780050075.