An Optimal Goal Programming Model to Recovery from Deadlocks

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

ABSTRACT
Process termination is a common strategy that is used to recover from deadlocks. However, terminating processes during their execution may affect and degrade the performance of the underlying system. The proposed solution in this paper is to select particular processes that can reduce the potential consequences of process termination in order to be terminated. A goal programming (GP) model is constructed to identify and select the best processes that can break a deadlock at lowest consequences of process termination. Several experimental tests are performed and the results showed that the proposed solution maintains the performance of the system during deadlock recovery compared to the other related methods.

Keywords
Goal Programming, Deadlock Recovery, Process Termination, Termination Cost.

1. INTRODUCTION
A deadlock in a computer system is a situation in which several processes are waiting infinitely to allocate some resources that are allocated by other waiting processes and cannot complete its tasks [1]. As a result, such situation can prevent the system from running and achieve its designated goals. Deadlock is an important issue that needs to be handled in computer systems, and in operating systems (OS) particularly [2-3]. Among the common strategies of deadlock handling, deadlock detection and recovery is accepted as a main feasible solution due to its applicable principles about resource allocation requirements of the processes [4-9].

Recovery from a deadlock can be performed when a deadlock occurs by terminating all or some deadlocked processes and releasing its allocated resources to be available for use [10]. However, aborting processes is found to be challenging because a great cost might result; utilizing of several allocated resources might be interrupted and part of computations and performed work might be wasted which negatively impacts the system performance [11]. Several desirable performance attributes for the system, such as throughput and resource utilization might be influenced and degraded by terminating the processes [12]. Another issue is to determine which processes are able to break the deadlock and solve the problem. Processes holding different resources dynamically and specifying the best set of processes that allocate enough resources to break the deadlock is a difficult task in such environment.

The objective of this paper is to find an optimal set of processes that can break the deadlock at minimum potential cost. Processes that can break deadlocks and whose termination will cause the minimum cost need to be specified and selected for termination. Thus, the problem of this paper can be represented in the following question:

- How can an optimal set of processes whose termination causes less impact on the overall system performance and can break a deadlock be selected?

Consequently, a goal programming (GP) model that seeks an optimal set of processes that can break the deadlock at minimum cost is proposed. Indeed, several selection criteria related to system performance and breaking deadlocks are proposed, and a GP model responsible for selecting processes that best achieve such criteria to be terminated is constructed. The remainder of this paper is organized as follows: the related works are reviewed in section 2. Section 3 presents the requirements and a detailed design of the proposed solution. Evaluation result and discussion is introduced in section 4. Finally, some conclusions are given in section 5.

2. RELATED WORKS
Several varied strategies are used for recovery from a deadlock [11] [13] [14]. Process termination, especially partial termination, is one common solution that is used for deadlock recovery [3,15]. In partial termination, one or more of the deadlocked processes are aborted and its resources are released until the deadlock is broken [16].Nevertheless, the primary issue is to find the best processes whose termination will result in minimum cost to be terminated [7-9, 17].

Many researchers have found out that the primary step of deadlock recovery is to select a process and then to abort it [18-20]. Different attributes of processes or resources have been adopted when selecting a process for termination. The simplest method is to abort any deadlocked process [21-22], or randomly terminate any process until a deadlock is broken [14,23]. The authors in [24] discuss minimum cost recovery for multiprogramming by selecting processes based on the number of held and requested resources; in addition to a number of requests that a process can satisfies. The cost represent the computation complexity in terms of time.

Another method proposed in [25] selects processes based on a dynamic process priority that was assigned based on several attributes such as: process age, process history, process code size, process priority, resource utilized, number of in/out-degree edges, and cycle participation. The author in [26] proposes the use of heuristics for choosing the right process such as the last process that has been loaded, or the process with the least remaining processing time. Many other criteria are used to guide process selection for termination such as process code size [27], number of deadlock cycles that the process involved [15,28], number of submitted operations [29,30], number of holding resources [9,28], number of terminated processes [31-32], process age [32-33] and process priority [8-9,20,34-35].

However, the cost of aborting processes has been mainly measured in terms of either time complexity [9,15,30,36], or message complexity [6,8,37]. Process termination may lead to a significant degradation in system performance (e.g.

throughput and resource utilization) rather than time and message complexity. The investigation or measurement of potential cost of process termination as an impact on system performance is generally missing in the literature. As pointed out in [14], however, cost is a general term and several factors may affect the decision of which deadlocked process should be terminated. Therefore, measuring and minimizing the cost of process termination in terms of system performance is required and adopted in this paper.

In fact, some process selection methods aimed to improve a specific performance aspect instead of maintaining overall system performance. In this sense, one method can be ideal in one aspect of system performance, but not in other aspects. Successful recovery from a deadlock is the one that maintains the overall system performance encompassing several desirable performance aspects simultaneously. Furthermore, such methods do not actually model a recovery strategy from detected deadlocks. Several desirable performance parameters that could improve the overall system performance will be considered in this paper when measuring and minimizing cost of process termination.

Moreover, little attention has been given to recovery from deadlocks in systems that have multiple instances for each resource type in in the literature. Deadlock detection and recovery in systems that have single instance of each resource type can be simply performed using a wait-for-graphs [13,15] which require less operations and overhead. In contrast, deadlock detection and recovery in systems that have multiple instances for each resource type is more challenging and the cost might be higher. Therefore, the proposed solution is intended to deal with resources that has multiple instances.

Consequentially, previous solutions of recovery from deadlocks depend on a deadlock detection algorithm to determine whether the deadlock is resolved, and they are not concerned with the ability to break deadlock in their selection strategy. Such a procedure requires frequent execution of deadlock detection algorithm for each occurrence of a deadlock situation. Consequently, valuable system resources may be wasted as a result of frequent execution of deadlock detection algorithm [13]; which increase the overhead and cost. However, the proposed solution do not require a detection algorithm to insure that a deadlock situation is resolved. Alternatively, the GP selected the processes according to its ability to break the deadlock. This can reduce the overhead of detection that might be required after each process selection and improve the performance.

3. THE PROPOSED SOLUTION

The proposed solution for minimum cost deadlock recovery is to determine which deadlocked processes should be terminated based on its ability to break the deadlock, and based on the potential cost that might be resulted from aborting such processes. Simultaneous achievement of such constraints is required when performing the selection. The following subsections present the details of the solution.

3.1 Breaking a Deadlock

A deadlock occurs when there is not enough available resources that can be used by waiting processes. Each deadlocked process awaits for resources that are occupied by other waiting processes. The goal of process termination is to release a number of resources that can satisfy the requests of one or more other waiting processes and leads to break the deadlock. Thus, the optimal set of processes holding resources that are enough to satisfy the requests of some deadlocked processes need to be specified and selected for termination. However, the proposed GP will be formulated to find a best set of processes that hold enough number of resources to break a deadlock to be terminated. This formulation guarantees recovery from the deadlock without need to use of deadlock detection algorithm.

3.2 Cost of Recovery from a Deadlock

Process termination may lead to interrupt resource usage, and to waste performed work and tasks for the terminated processes which eventually affect the overall system performance, especially throughput and resource utilization [11,14,25]. Thus, several cost factors - as pointed out in [14] - that might contribute to maintain the system performance will be used in the proposed solution to measure and minimize the cost of aborting processes. These factors will be modeled and formulated in this paper to be used as selection criteria that guide to a minimum cost deadlock recovery. The cost factors are presented in Table 1.

Table 1. Cost Factors (selection criteria) for Process Selection [14]

1. What is the priority of the process
2. How long time the process has computed
3. How long time the process still requires in order to complete
4. How many resources the process has used
5. How many resources left for the process to complete

Indeed, each factor can contribute in minimizing the cost by maintaining a specific performance parameter for the system. For example, reducing the number of wasted resources is desired and needs to be attained when recovering from a deadlock; thus the number of resources that the process has used needs to be considered as a cost factor in order to be minimized. Using the previous cost factors can contribute mainly in maintaining throughput, resource utilization and fairness among process, which are important and desired attributes in resource sharing and process cooperating system environments [1, 12]. Such performance attributes might be significantly degraded when recovering from deadlocks by process termination.

Fairness among processes can be achieved by selecting processes according to a predetermined priority rather than any other arbitrary reference. A system that achieves fairness among processes is better in terms of cost. Throughput can be maintained by either increasing the completed work or tasks in a given time or increasing the number of processes completed per unit time [1]. Indeed, processes that require more execution time and higher number of resources in order to complete are assumed to consume more resources and finish the execution much later, and terminating such processes can let smaller processes to finish in a given time which attain higher throughput. Furthermore, processes that have computed longer time are supposed to complete more work or tasks, thus keeping such processes and terminating processes that have computed less time can help in maintaining throughput. Moreover, higher performed work and completed tasks can indicate higher resource utilization. Thus, terminating processes that hold and use less number of resources reduce the number of wasted resources that are already being used, and keep more resources executing and completing their designated tasks rather than interrupting their execution which indicates better resource utilization. A system that achieves higher throughput and resources utilization is better at a lower cost.

3.3 The Goal Programming Model

The objective of this paper is to perform deadlock recovery at minimum cost by process termination. Thus, cost factors in addition to the ability of breaking the deadlock need to be used as selection criteria, and the selection of processes for termination will be performed up on these criteria. However, the challenge lies in finding a solution that best achieves such criteria concurrently; processes that are optimal in minimizing cost may not guarantee to break the deadlock and vice versa. Furthermore, a set of processes that is optimal in one cost factors as a selection criterion may not be optimal in others. The best solution in such case is always a compromise, and all metrics of interest must be taken into account concurrently.

Accordingly, a GP model responsible for finding an optimal set of processes that best satisfies such multiple selection criteria is proposed. GP is a multi-criteria satisfying methodology that seeks a solution that best fits or satisfies the desired set of criteria in a multi criteria decision making problems [38].

In fact, the main objective function to be formulated in the GP model is to minimize the cost that has several factors. Consequently, percentage of execution time (*T*), percentage of resource allocation (*R*), and priority violation (P_r) are proposed as input variables for the GP model to help measure and minimize the cost that represents the proposed factors. *T*, *R*, and P_r variables can be calculated for each deadlocked processes based on available information that can be obtained dynamically from the underlying operating system at a time of performing deadlock detection and recovery. Through such variables, the proposed GP can find processes that best satisfy the adopted cost factors and minimize the cost. Table 2 shows the description of the proposed formulation variables and its relation to the cost factors.

Table 2. Information about Formulation Variables (V: variable and CF: cost factor as in Table 1)

V	CF	Parameter	Description	GP selection
P_r	1	Priority	Priority of the process.	Lower value
T	2	Time computed	Amount of time that the process has computed.	Lower value
	3	Time required	Amount of remaining time that required for the process in order to complete.	higher value
R	4	Resource allocation	Number of resources of each type currently allocated to each process.	Lower value
	5	Resource required	Number of resources that required for each process in order to complete.	higher value

T, *R*, and P_r variables can be calculated for a process *i* using the following equations.

$$R_i = \frac{\text{Resource allocation}_i}{\text{Resource allocation}_i + \text{Resource required}_i} \quad (1)$$

$$T_i = \frac{\text{Time computed}_i}{\text{Time computed}_i + \text{Time require d}_i} \quad (2)$$

$$P_{ri} = \text{priority}_i - \min(\text{priority}) \quad (3)$$

Where Min (priority) is a function that returns smallest value of priority for all deadlocked processes that exist in the system at a time of performing deadlock recovery.

As a result, minimizing the cost of aborting processes as a main objective can be divided into the following sub-goals as: (i) minimize the percentage of wasted execution time *(T)*; (ii) Minimize percentage of wasted resources*(R)*; (iii) Minimize priority violation (P_r).

Additionally, in order to select a set of processes that able to break the deadlock, set of related constraints are developed and used in the model. Finally, the following GP model is constructed as presented in Figure 1:

1 Min $Cost$;
2 Subject to:
3 $\text{Cost} = d_R^- + d_R^+ + d_T^- + d_T^+ + d_P^- + d_P^+$;
4 $(\sum_{i=1}^{n} R_i * P_i) - d_R^- + d_R^+ = 0$;
5 $(\sum_{i=1}^{n} T_i * P_i) - d_T^- + d_T^+ = 0$;
6 $(\sum_{i=1}^{n} P_{r_i} * P_i) - d_P^- + d_P^+ = 0$;
7 For k=0… n-1
8 For j=1 … r
9 $(\sum_{i=0}^{n-1} \text{request}_{i,j} * T_k P_i) \leq available_j + (\sum_{i=0}^{n-1} hold_{i,j} * P_i) + \sum_{m=0,k\neq 0}^{k-1} (\sum_{i=0}^{n-1} hold_{i,j} * T_m P_i)$;
10 For i=0… n-1
$(\sum_{k=0}^{n-1} T_k P_i) + P_i = 1$

Fig. 1 the proposed GP model for recovery from deadlocks

Where *n* is the number of deadlocked processes, *r* is the number of resource types in the system, $\text{request}_{i,j}$ is the number of requested resources of type *j* by a process *i*, $hold_{i,j}$ is the number of allocated resources of type *j* by a process *i*, P_i and $T_k P_i$ are binary decision variables (0/1 variables), and $d_R^-, d_R^+, d_T^-, d_T^+, d_P^-, d_P^+$ are deviational variables. The expected output of the model is a vector of 0/1 values corresponds to each process decision variable (P_i). 0 means that the correspondent process is not selected, where 1 means that the associated process is suitable for termination. Processes with 1 value are the optimal processes whose termination together as a set can break the deadlock at minimum cost.

4. EVALUATION RESULTS AND DISCUSSION

In order to measure the ability of the proposed GP model to perform deadlock recovery with the least possible cost, and to compare its performance with related methods, a simulation system was constructed and several tests were run. The simulation was executed to generate varied deadlock situations under the following assumptions:

- The number of allocated resources, the number of requested resources, the amount of computed time, the amount of required time, and the priority for each process were selected randomly.
- The simulation selected varied numbers of resource types **(*r*)** with random number of instances in each type.
- Each deadlocked process is holding one or more resources in the system.
- Each deadlocked process requests a number of resources less than or equal to the number the system has.
- All system resources are in use
- The number of requested resources were selected under the constraint $\boldsymbol{request_{i,j}+hold_{i,j}}$ less than or equal to all resources of that type in system.

However, deadlock situations were generated randomly under three different values of resource types, where ***r*** = 1, 4, and 8.

The number of requested resources by each process were selected randomly between 0 and MAXrequest. MAXrequest can reflect the relation between the number of allocated and requested resources to the number of all resources in the system. MAXrequest = factor*($request_i$-$hold_i$). The factor reflects the degree of how costly the recovery from a deadlock [24]. The selected factor values were 0.1, 0.5, and 1.0. For each of the 9 combinations of $\boldsymbol{r}$ and factor values, priorities were generated randomly between 0 and MAXpriority where MAXpriority= 2, 10, and 20, respectively. This makes up three variations of priorities. For each of the 27 combinations of resource types, factors, and MAXpriority values, 250 deadlock situation were generated resulted in a total of 6750 deadlock situations.

In sum, all scenarios have been classified, summarized, and tested as depicted in Table 3 into three sets: Set (A) is to measure and compare the performance of the related methods in handling deadlocks with different number of resource types, set (B) is to measure and compare the performance of the related methods in handling deadlocks with different values of deadlock factors, and set (C) is to measure and compare the performance of the related methods in handling deadlocks with different variations of priorities.

Table 3. Simulation Scenarios

No	Resource type		Factor		Priority	
	Test	Value	Test	Value	Test	Value
1	$(A)_1$	1	$(B)_{0.1}$	0.1	$(C)_{0\text{-}2}$	0-2
2	$(A)_4$	4		0.1		0-2
3	$(A)_8$	8		0.1		0-2
4	$(A)_1$	1	$(B)_{0.5}$	0.5		0-2
5	$(A)_4$	4		0.5		0-2
6	$(A)_8$	8		0.5		0-2
7	$(A)_1$	1	$(B)_{10}$	1		0-2
8	$(A)_4$	4		1		0-2
9	$(A)_8$	8		1		0-2
10	$(A)_1$	1	$(B)_{0.1}$	0.1	$(C)_{0\text{-}10}$	0-10
11	$(A)_4$	4		0.1		0-10
12	$(A)_8$	8		0.1		0-10
13	$(A)_1$	1	$(B)_{0.5}$	0.5		0-10
14	$(A)_4$	4		0.5		0-10
15	$(A)_8$	8		0.5		0-10
16	$(A)_1$	1	$(B)_{10}$	1		0-10
17	$(A)_4$	4		1		0-10
18	$(A)_8$	8		1		0-10
19	$(A)_1$	1	$(B)_{0.1}$	0.1	$(C)_{0\text{-}20}$	0-20
20	$(A)_4$	4		0.1		0-20
21	$(A)_8$	8		0.1		0-20
22	$(A)_1$	1	$(B)_{0.5}$	0.5		0-20
23	$(A)_4$	4		0.5		0-20
24	$(A)_8$	8		0.5		0-20
25	$(A)_1$	1	$(B)_{10}$	1		0-20
26	$(A)_4$	4		1		0-20
27	$(A)_8$	8		1		0-20

Consequently, for each deadlock situation generated, the deadlock was recovered by terminating processes that selected using five methods namely; least required resources (LRR), least required processing time (LRPT), first detected first aborted process (order of detection), priority, and the proposed GP model. Each method selects a set of processes to be aborted and recover from each deadlock situation, and the cost was calculated based on the selected processes.

Figure 2 shows the performance of the GP model compared to the average performance of all other related methods under different system parameters. Firstly, the results showed that performing deadlock recovery becomes more difficult with the increase of the values of parameters. That is, the expected cost of deadlock recovery increases with environmental changes. Secondly, the average performance of the GP model is significantly better than average performance of the four related methods and it can scale to varying environmental changes at a minimum cost.

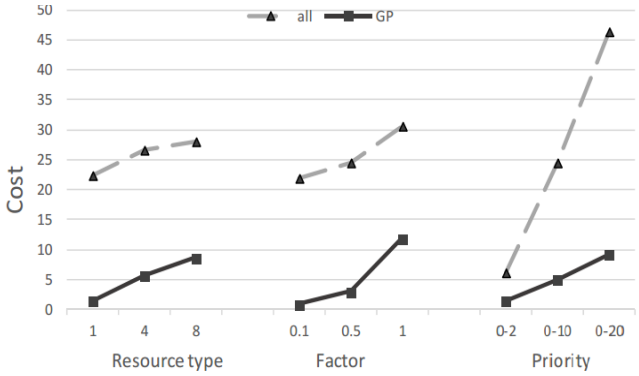
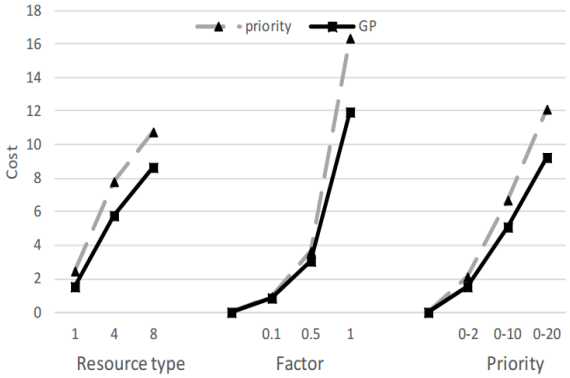
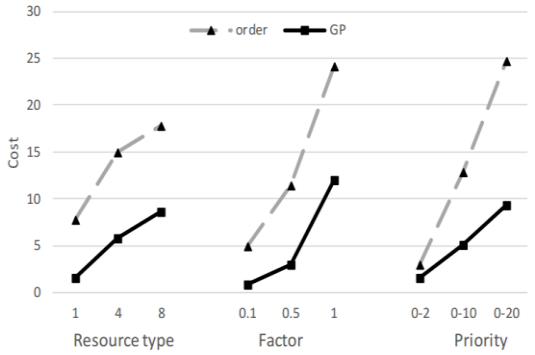


Fig. 2 avarage cost resulted when using GP compared to avarge cost of all other methods(LRR,LRPT,priority and order)

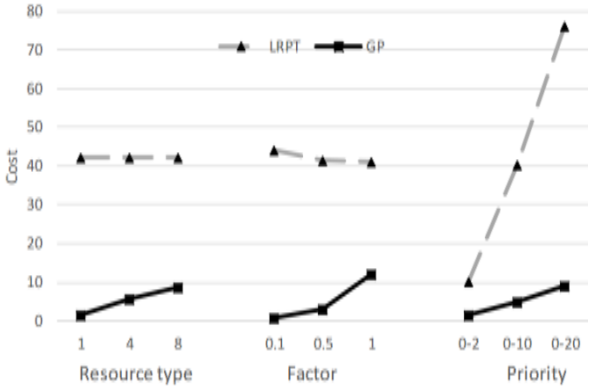
More specifically, as depicted in Figure 3 (a-d), the proposed GP method performs better than the other methods and recover all deadlock situations at minimum cost under all tests. This could be explained as the GP selects processes based on the potential cost that might be resulted and based on its ability to break deadlock simultaneously. Furthermore, GP seeks processes that best achieves all cost factors which have better result for the overall system performance. Comparatively, methods such as order selection do not take into account any selection criteria which significantly increased the recovery cost. Other methods that are based on specific criteria such as the number of required resources, remaining processing time, or priority consider only one cost factor which partially reduces the cost.



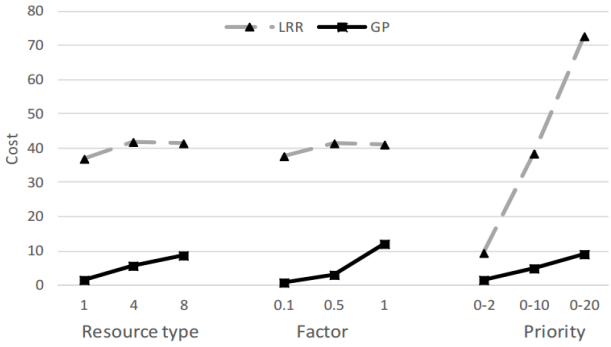
(a) avarage cost resulted when using GP compared to avarge cost resulted when using priority method



(b) avarage cost resulted when using GP compared to avarge cost resulted when using order method



(c) average cost resulted when using GP compared to avarge cost resulted when using LRPT method



(d) avarage cost resulted when using GP compared to avarge cost resulted when using LRR method

Fig. 3 Performance of the GP compared to each related method

Figure 4 summarizes the overall cost achieved by all methods for all generated deadlock situations. The cost of aborting processes resulted when using the GP method was the lower among the other methods with varying degrees. The GP outperforms the LRR with 86% efficiency, LRPT with 87% efficiency, order with 60% efficiency, and priority with 24% efficiency. Furthermore, the detection algorithm was not required when performing recovery by the proposed GP model which significantly reduces the recovery cost in terms of overhead. In contrast, the detection algorithm was frequently executed for each deadlock situation when it was recovered by each other method.

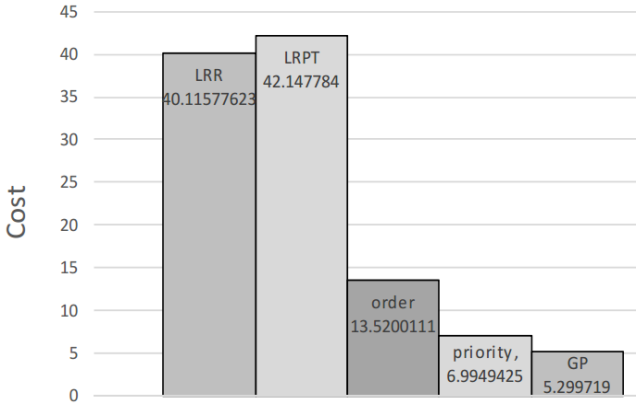


Fig. 4 Overall cost achived by all methods for all generated dealock situations

5. CONCLUSION

Deadlock recovery at minimum cost has been addressed in this paper. The proposed solution was to find and terminate set of processes that best reduce the potential termination cost. Performance attributes such as throughput, resource utilization and fairness are used as selection criteria in addition to the ability to break the deadlock to help in choosing processes for termination. Accordingly, a linear goal programming model was proposed to find processes that best satisfy such multiple selection criteria. Measuring and minimizing recovery cost based on the proposed criteria reduced the negative impact of aborting processes on the overall system performance.

The performance of the proposed GP has been measured and compared to some related methods that might be used to select a process for termination. The result showed that the GP selected the best set of processes for termination that were able to break the deadlock and conducted minimum potential cost. The GP outperforms the other selection methods with 64% average overall efficiency. However, as a future development of the new model, the proposed selection criteria can be dynamically weighed according to their level of achievement in the system; and use these weights in the proposed GP model. This might best maintain such desirable performance criteria dynamically.

6. REFERENCES

[1] A. S. Tanenbaum, Modern operating systems (Prentice Hall Press, 2014).

[2] Y. Nir-Buchbinder, R. Tzoref, S. Ur, Deadlocks: From exhibiting to healing. In Runtime Verification, LNCS 5289, Springer Berlin Heidelberg, 2008, 104-118.

[3] V. S. Kondhalka, Deadlock detection and recovery in Linux. Ph.D. Thesis, San Diego State University, 2012.

[4] Izumi, A., Dohi, T., & Kaio, N. Deadlock Detection Scheduling for Distributed Processes in the Presence of System Failures. Proceedings of the 6th Pacific Rim International Symposium on Dependable Computing IEEE (page 133, 2010).

[5] E. Knapp, Deadlock detection in distributed databases. ACM Computing Surveys (CSUR), Vol. 19, n. 4, pp. 303-328, 1987.

[6] S. Lee, J. L. Kim, Performance analysis of distributed deadlock detection algorithms, Knowledge and Data Engineering, IEEE Transactions on, Vol. 13, n. 4, pp. 623-636, 2001.

[7] M. Singhal, Deadlock detection in distributed systems. IEEE Computer, Vol. 22, n. 11, pp. 37-48, 1989.

[8] J. R. González de Mendívil, F. Fariña, J. R. Garitagotia, C. F. Alastruey, J. M. Bernabeu-Auban, A distributed deadlock resolution algorithm for the AND model. IEEE Transactions on Parallel and Distributed Systems, Vol. 10, n. 5, pp. 433-447, 1999.

[9] I. Terekhov, T. Camp, Time efficient deadlock resolution algorithms. Information Processing Letters, Vol. 69, n. 3, pp. 149-154, 1999.

[10] R. C. Holt, Some deadlock properties of computer systems. ACM Computing Surveys (CSUR), Vol. 4, n. 3, pp. 179-196, 1972.

[11] K. S. Vaisla, M. Goswami, A. Singh, VGS Algorithm-an Efficient Deadlock Resolution Method. Journal of Computer Applications, Vol. 44, n. 1, pp. 29-33, 2012.

[12] S. M. Darwish, A. A. El-Zoghabi, M. H. Hassan, Soft Computing for Database Deadlock Resolution. International Journal of Modeling and Optimization, Vol. 5, n. 1, pp. 15, 2015.

[13] F. Zeng, Just-in-time and just-in-place deadlock resolution. Ph.D. Thesis, New Brunswick Rutgers, The State University of New Jersey, 2007.

[14] A. Silberschatz, P. B. Galvin, G. Gagne, Operating system concepts (Wiley, 2013).

[15] Chow, Y. C., Kostermeyer, W. F., & Luo, K. Efficient techniques for deadlock resolution in distributed systems, Proceedings of the Fifteenth Annual International in Computer Software and Applications Conference, IEEE, (Page: 64 Year of Publication: 1991).

[16] Y. Ling, S. Chen, C. Y. Chiang, On optimal deadlock detection scheduling. IEEE Transactions on Computers, Vol. 55, n. 9, pp. 1178-1187, 2006.

[17] Macri, P. P. Deadlock detection and resolution in a CODASYL based data management system. In Proceedings of the 1976 ACM SIGMOD international conference on Management of data, ACM, (Page: 45 Year of Publication: 1976).

[18] Villadangos, J., Fariña, F., Cordoba, A., de Mendivil, J. R. G., & Garitagoitia, J. R. Knot resolution algorithm and its performance evaluation. In Proceedings of the Conference on Parallel, Distributed and Network-Based Processing, IEEE (Page: 227 Year of Publication: 2003).

[19] Lee, S. Fast detection and resolution of generalized distributed deadlocks. Proceedings of the10th Euromicro Workshop on Parallel, Distributed and Network-based Processing, IEEE, (Page: 429 Year of Publication: 2002).

[20] Cordoba, A., Fariña, F., Garitagoitia, J. R., de Mendivil, J. R. G., & Villadangos, J. A low communication cost algorithm for distributed deadlock detection and resolution. Proceedings of the Eleventh Euromicro Conference on Parallel, Distributed and Network-Based Processing, IEEE, (Page: 235 Year of Publication: 2003).

[21] Mitchell, D. P., & Merritt, M. J. A distributed algorithm for deadlock detection and resolution. In Proceedings of the third annual ACM symposium on Principles of distributed computing , ACM, (Page: 282Year of Publication: 1984).

[22] M. Roesler,W. Burkhard, A. Resolution of deadlocks in object-oriented distributed systems. Computers, IEEE Transactions on, Vol. 38, n. 8, pp. 1212-1224, 1989.

[23] Hashemzadeh, M., Farajzadeh, N., & Haghighat, A. T. Optimal detection and resolution of distributed deadlocks in the generalized model. In Proceedings of the 14th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing,IEEE, (Page: 4-pp Year of Publication: 2006).

[24] J. T. Leung., E. K.Lai, On minimum cost recovery from system deadlock. IEEE Transactions on Computers, Vol. 28, n. 9, pp. 671-677, 1979.

[25] V. Geetha, N. Sreenath, Performance Analysis of Victim Selection Algorithms in Distributed Systems and Proposal of Weight Based Resolution Strategy. International Journal of Computer Science, Vol. 2, n. 4, pp. 40-44, 2012.

[26] Schruben, L. W. Deadlock detection and avoidance in cluster tools. In Proceedings of the 1999 International Conference on Semiconductor Manufacturing Operational Modeling and Simulation, (Page: 31 Year of Publication: 1999).

[27] Weikum, G., & Vossen, G. Transactional information systems. ACM, 2002

[28] P. Chahar, S.Dalal, Deadlock Resolution Techniques: An Overview.International Journal of Scientific and Research Publications, Vol. 3, n. 7, pp. 1-5, 2013.

[29] Lin, X., Orlowska, M. E., & Zhang, Y. An optimal victim selection algorithm for removing global deadlocks in multidatabase systems. Proceedings In TENCON'94. IEEE Region 10's Ninth Annual International Conference. Theme: Frontiers of Computer Technology, (Page: 501 Year of Publication: 1994).

[30] Lin, X., & Chen, J. . An optimal deadlock resolution algorithm in multidatabase systems. Proceedings in the International Conference on In Parallel and Distributed Systems IEEE, (Page: 516 Year of Publication: 1996).

[31] Sapra, P., Kumar, S., & Rathy, R. K. Deadlock Detection and Recovery in Distributed Databases. International Journal of Computer Applications, Vol. 73, n. 1, pp. 32-36, 2013.

[32] R. Agrawal, M. J. Carey, L. W. McVoy, The performance of alternative strategies for dealing with deadlocks in database management systems. IEEE Transactions on Software Engineering, Vol. SE-13, n. 12, pp. 1348-1363, 1987.

[33] Alom, M., Henskens, F., & Hannaford, M. Deadlock Detection Views of Distributed Database. Proceedings in the Sixth International Conference on Information Technology: New Generations,IEEE, (Page: 730 Year of Publication: 2009).

[34] Al Shayeji, M. H., Fairouz, A., & Samrajesh, M. D. An Enhanced Distributed Deadlock Detection and Recovery in Process Networks. International Journal of Computer and Electrical Engineering, Vol. 4, n. 3, pp. 298-302, 2012.

[35] Sinha, M. K., & Natarajan, N. A priority based distributed deadlock detection algorithm. IEEE Transactions on Software Engineering, Vol. 1, pp. 67-80, 1985.

[36] Chen, S., & Ling, Y. Stochastic analysis of distributed deadlock scheduling. In Proceedings of the twenty-fourth annual ACM symposium on Principles of distributed computing, (Page: 265 Year of Publication: 2005).

[37] S. Lee, Fast, centralized detection and resolution of distributed deadlocks in the generalized model. IEEE Transactions on Software Engineering, Vol. 30, n. 9, pp. 561-573, 2004.

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