IMPLEMENTATION OF ARTIFICIAL NEURAL NETWORK IN CONCURRENCY CONTROL OF COMPUTER INTEGRATED MANUFACTURING(CIM) DATABASE

P. Raviram ravirampedu@gmail.com
Research Scholar/Department of Computer Science and Engineering
Vinayaka Mission's University
Salem, 636 308, INDIA

R.S.D. Wahidabanu rsdwb@yahoo.com
Professor & Head
Department of Electronics and Communication Engineering
Government College of Engineering
Salem, 636 011, INDIA

ABSTRACT

Manufacturing database store large amount of interrelated data. The designers access specific information or group of information in the data. Each designer accessing an entity tries to modify the design parameters meeting the requirements of different customers. Sister concerns of the same group of company will be modifying the data as per design requirements. When information is updated with new modification by different group of designers, what is the order in which modification of the data has to be allowed. If simultaneous access of the information is done, how to maintain the consistency of the data. and a designer voluntarily corrupts the data, how to make sure the designer is responsible for the corruption of data. In any case if the transaction process corrupts the data, how to maintain the consistency of the data. Deleting the information wantedly can be identified with extra security for the data. However, when transaction protocol is not implemented properly, then corruption of data in the form of misleading information that showing less numerical value than what it has to be or showing more numerical than before updation. In this research work, we have proposed a neural network method for the managing the locks assigned to objects and the corresponding transactions are stored in a data structure. The main purpose of using the ANN is that it will require less memory in storing the lock information assigned to objects. We have attempted to use backpropagation algorithm for storing lock information when multi users are working on computer integrated manufacturing (CIM) database.

Keywords: Concurrency control, locks, backpropagation algorithm, neural network, CIM database, Knowledge management.

1. INTRODUCTION

Knowledge management in advanced database have been considered as an interesting research area in the recent past. Real-Time database systems (RTDB) and Active database systems have been discussed and implemented respectively to support non-traditional applications. Few researches concentrate on the integration of active and real-time database systems and is very much used in computer integrated manufacturing (CIM). New problems are evolved in concurrency control (CC)[4,13] of real-time database systems. Conventional CC protocols are more concerned about the serializability but real-time database systems also focus on transaction deadlines. The situation is more complicated when real-time databases integrated with active characteristic. RTDBs must not only respond to the external transactions but also the internal triggered events. Due to the triggering structure in RTDBs, a dynamic CC algorithm is needed. If we simply apply existing conventional database or real time database CC protocols, a lot of CPU processing time will be wasted and transactions may not be able to complete before their deadlines.

2. TRANSACTION PROCESS

Transaction is series of actions, carried out by user or application, which accesses or changes contents of database. It is a logical unit of work on the database. It transforms database from one consistent state to another, although consistency may be violated during transaction[2,3]. Concurrency is the process of managing simultaneous operations on the database without having them interfere with one another. It prevents interference when two or more users are accessing database simultaneously and atleast one is updating data. Although two transactions may be correct in themselves, interleaving of operations may produce an incorrect result. Three potential problems caused by concurrency are lost update, uncommitted dependency and inconsistent analysis. Executions of transactions guaranteed to ensure consistency is identified by the concept of serializability with those schedule of reads / write. Serial schedule is where operations of each transaction are executed consecutively without any interleaved operations from other transactions. Nonserial Schedule: Schedule where operations from set of concurrent transactions are interleaved techniques used for concurrency Control are Locking and Timestamping. Both are conservative approaches when delay transactions in case they conflict with other transactions. Optimistic methods assume conflict is rare and only check for conflicts at commit.

Transaction uses locks to deny access to other transactions and so prevent incorrect updates. A transaction must claim a shared (read) or exclusive (write) lock on a data item before read or write. Lock prevents another transaction from modifying item or even reading it, in the case of a write lock. Rules of locking are, if transaction has shared lock on item, can read but not update item, and if transaction has exclusive lock on item, can both read and update item, Reads cannot conflict, so more than one transaction can hold shared locks simultaneously on same item, Exclusive lock gives transaction exclusive access to that item.

3. TRANSACTIONS REQUIREMENTS IN CAD DATABASE

Design and development of a product (shown in Figure 1) is the first and foremost step in a manufacturing industry. This process is recurrent and repetitive until it reaches a final approved design and development stage. Design and development activity involves defining and describing the product, drawing the product in the computer using computer aided drafting(CAD) software making modifications in the drawing, proving suitable material combinations for the product, defining various sizes for the product, providing safety factor provision based on the end application, satisfying customer requirements. The entire process will be generally interactive between a designer and the customer with one to one direct contact, or interactive discussions between designers at various locations, or independent design decisions by various designers who are located at different places and are accessing the same database which is centralized and sometimes distributed. When many designers are involved in designing an object in the database a major problem of concurrency as well as version of the product developed occurs.

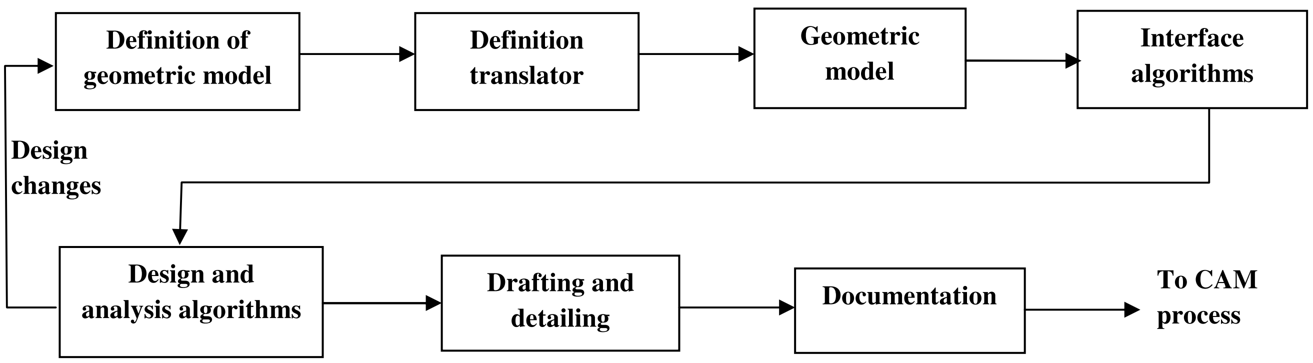

FIGURE 1: Schematic diagram of Design and development.

Majority of transaction will be done with long time gap. In the existing commercial database, all the equations and procedures are already coded with all the rules required which may change time to time.[12] In such case, whenever a user is accessing the data in the form of read / write the transaction quickly. He may not choose the choice of interest charged for the cash he swiped. using his credit card. This indicates transaction with minimum amount of time.

Transaction with longer time period : In case of computer design process even though design formulae have been encoded at the time of software development, the users will have their choice of choosing their design requirements[1,2]. They will do design based on their previous experiences and on the requirements of customers, based on the availability of machine capabilities in the workshop. The activity of deciding the optimum design will take long time for completing the transaction. Suppose, a similar design is done by another user very quickly, which may be based on some criteria, the question is whether the second person can be permitted to update the database or should he have to wait for the first person to the design process. What type of transaction concept that has to be adopted is based on many criteria which may not be readily fixed.

Controlling transaction with users choice: Most of the CAD transactions are based on interaction among many users. Atleast two users would transact the knowledge, discuss and come to a conclusion whether such design can be finalized. In such case, the final commit in the database should be possible. This should not become impossible because of basic transaction rules.

Cooperating the views of the designers in synchronization Many users working on shared objects cannot be serialized. The shared objects shall pass among them when modifying two parts of the same object parallelly to create new version of the object. In this case complete serializability is not possible. New concept is being developed to meet maximum seriazability in such type of shared interactive design environment.

4. EARLIER APPROACHES TO GROUP TRANSACTIONS

Developers of large project often work in small teams. A policy is to define the kinds of interactions allowed among members of the same team as opposed to interactions between teams. A group paradigm to deal with consistency of replicated data in an unreliable distributed system[3]. We can hierarchically divide the problem of achieving serializability into two simpler ones: (1) a local policy that ensures a total ordering of all transactions within a group; and (2) a global policy that ensures correct serialization of all groups. Groups, like nested transactions, is an aggregation of a set of transactions[5]. There are differences between groups and nested transactions. A nested transaction is designed *a priori* in a structured manner as a single entity that may invoke subtransactions, which may themselves invoke other subtransactions. Groups do not have any *a priori* assigned structure and no predetermined precedence ordering imposed on the execution of transactions within a group. Another difference is that the same concurrency control policy is used to ensure synchronization among nested transactions at the root level and within each nested transaction.. The group paradigm was introduced to model inter-site consistency in a distributed database system. It can be used, to model teams of developers, where each team is modeled as a group with a local concurrency control policy that supports synergistic cooperation.

CAD Transactions in Groups : The group-oriented model does not use long-lived locks on objects in the public database. The conversational transactions model sets long-lived locks on objects that checked out from the public database until they are checked back into the public database. The group-oriented model categorizes transactions into *group transactions* (GT) and *user transactions* (UT). Any UT is a subtransaction of a GT. The model also provides primitives to define groups of users with the intention of assigning each GT a user group. Each user group develops a part of the project in a *group database*. A GT reserves objects from the public database into the group database of the user group it was assigned. Within a group database, individual designers create their own user database, and they invoke UTs to reserve objects from the group database to their user database.

In the group-oriented model, user groups are isolated from each other. One user group cannot see the work of another user group until the work is deposited in the public database. Group transactions are thus serializable. Within a group transaction, several user transactions can run concurrently. These transactions are serializable unless users intervene to make them cooperate in a non-serializable schedule. The basic mechanism provided for relaxing serializability is a version concept that allows parallel development (branching) and notification. Versions are derived, deleted, and modified explicitly by a designer only after being locked in any one of a range of lock modes.

The model supports five lock modes on a version of an object: (1) read-only, which makes a version available only for reading; (2) read/derive, which allows multiple users to either read the same version or derive a new version from it; (3) shared derivation, which allows the owner to both read the version and derive a new version from it, while allowing parallel reads of the same version and derivation of different new versions by other users; (4) exclusive derivation, which allows the owner of the lock to read a version of an object and derive a new version, and allows only parallel reads of the original version; and (5) exclusive lock, which allows the owner to read, modify and derive a version, and allows no parallel operations on the locked version. Using these lock modes, several designers can cooperate on developing the same design object. The exclusive lock modes allow for isolation of development efforts (as in traditional transactions), if that is what is needed. To guarantee consistency of the database, designers are only allowed to access objects as part of a transaction. Each transaction in the group-oriented model is two-phase, consisting of an acquire phase and a release phase. Locks can only be strengthened (i.e., converted into a more exclusive mode) in the acquire phase, and weakened (converted into a more flexible lock) in the release phase. If a user requests a lock on a particular object and the object is already locked with an incompatible lock, the request is rejected and the initiator of the requesting transaction is informed of the rejection. This avoids the problem of deadlock, which is caused by blocking transactions that request unavailable resources. The initiator of the transaction is notified later when the object he requested becomes available for locking. In addition to this flexible locking mechanism, the model provides a read operation that breaks any lock by allowing a user to read any version, knowing that it might be about to be changed. This operation provides the designer (more often a manager of a design effort) the ability to observe the progress of development of a design object, without affecting the designers doing the development.

5. BACKPROPAGATION ALGORITHM (BPA)

An artificial neural network(ANN)[7] is an abstract simulation of a real nervous system that contains a collection of neuron units, communicating with each other via axon connections. Such a model bears a strong resemblance to axons and dendrites in a nervous system. Due to this self-organizing and adaptive nature, the model offers potentially a new parallel processing paradigm. This model could be more robust and user-friendly than the traditional approaches. ANN can be viewed as computing elements, simulating the structure and function of the biological neural network. These networks are expected to solve the problems, in a manner which is different from conventional mapping. Neural networks are used to mimic the operational details of the human brain in a computer. Neural networks are made of artificial 'neurons', which are actually simplified versions of the natural neurons that occur in the human brain. It is hoped, that it would be possible to replicate some of the desirable features of the human brain by constructing networks that consist of a large number of neurons. A neural architecture (shown in Figure 2) comprises massively parallel adaptive elements with interconnection networks, which are structured hierarchically.

The BPA[6] uses the steepest-descent method to reach a global minimum. The number of layers and number of nodes in the hidden layers are decided. The connections between nodes are initialized with random weights. As shown in Table 1 a pattern from the training set is presented in the input layer of the network and the error at the output layer is calculated. The error is propagated backwards towards the input layer and the weights are updated. This procedure is repeated for all the training patterns. At the end of each iteration, test patterns are presented to ANN, and the classification performance of ANN is evaluated. Further training of ANN is continued till the desired classification performance is reached.

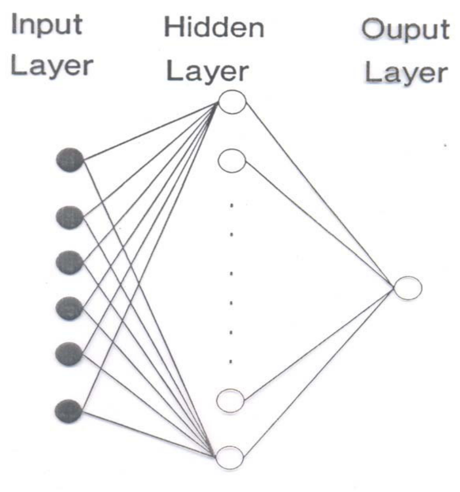

FIGURE 2: Multilayer perception.

6. INTELLIGENT LOCKING STRATEGY PROPOSED LOCKING IMPLEMENTATION

Inbuilt library functions for the bolt are available in standard CAD[8, 9,10,11, 12] softwares. In Figure 3, the bolt and its entities are shown schematically. The bolt drawing (Figure 3) is used to manufacture one variety of bolt that has to be used to clamp two plates. This drawing file will be accessed by many designers who will choose their choice of designs and the designs are stored in the same location of the server. Each designer can choose their option of changing the shaped , dimension of different components of the subassembly of the bolt. When any modification is done for one subassembly, due to associative dimensioning concept, the dimensions of the entire bolt shape and dimensions can change.

Problems faced in storing the modified drawing. If it is associative dimensioning, then all the changes in shape and size of the bolt system have to be updated for a small change in the dimension of the subassembly. At the same time, another designer would want to retain earlier version. of the drawing.

When more than one user adopts similar changes in the file, and when they commit at the same time, how to maintain the consistency of the bolt file.

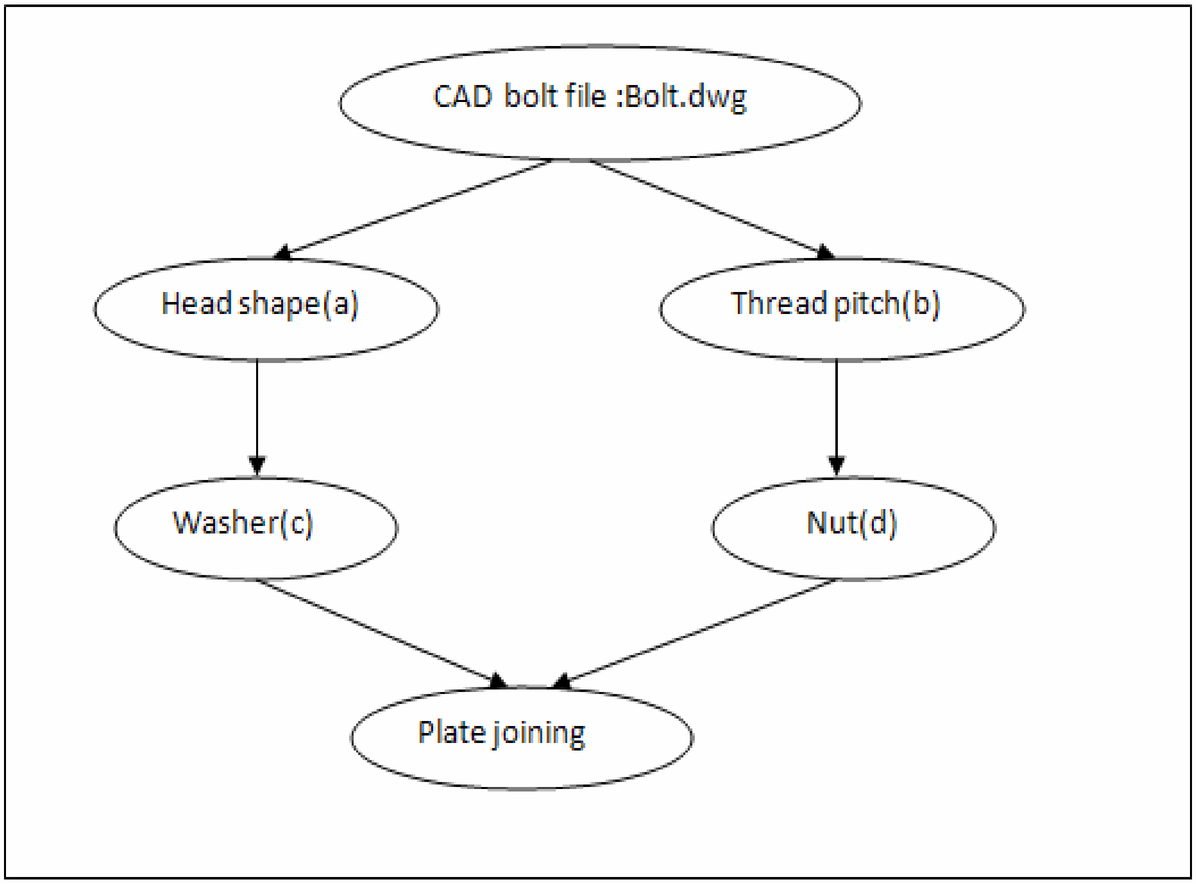

FIGURE 3: A bolt file contains entities.

STEPS INVOLVED.

FORWARD PROPAGATION

The weights and thresholds of the network are initialized.

The inputs and outputs of a pattern are presented to the network.

The output of each node in the successive layers is calculated.

o(output of a node) = $1/(1+\exp(\sum w_{ij} x_i + \Theta))$

The error of a pattern is calculated

$E(p) = (1/2) \sum(d(p) - o(p))^2$

REVERSE PROPAGATION

The error for the nodes in the output layer is calculated

δ(output layer) = o(1-o)(d-o)

The weights between output layer and hidden layer are updated

$W(n+1) = W(n) + \eta\delta$(output layer) o(hidden layer)

The error for the nodes in the hidden layer is calculated

δ(hidden layer) = o(1-o) $\sum\delta$(output layer) W(updated weights between hidden and output layer)

The weights between hidden and input layer are updated.

$W(n+1) = W(n) + \eta\delta$(hidden layer) o(input layer)

The above steps complete one weight updation

Second pattern is presented and the above steps are followed for the second weight updation.

When all the training patterns are presented, a cycle of iteration or epoch is completed.

The errors of all the training patterns are calculated and displayed on the monitor as the mean squared error(MSE).

$E(MSE) = \sum E(p)$

TABLE 1: Back-propagation algorithm.

In Table 2 defines the variables used for training the ANN about locks assigned to different objects.

User	Object	mode

TABLE 2: Variables considered for training and testing of ANN for lock management.

where
User represents the client
Object represents the entire Manufacturing related file or an entity in the file
Mode represents type of lock assigned to an object.

exclusive *(X) mode*. Data item can be both read as well as written.
shared *(S) mode*. Data item can only be read..
intention-shared (IS): indicates explicit locking at a lower level of the tree but only with shared locks.
intention-exclusive (IX): indicates explicit locking at a lower level with exclusive or shared locks
shared and intention-exclusive (SIX): the subtree rooted by that node is locked explicitly in shared mode and explicit locking is being done at a lower level with exclusive-mode locks.
A intention locks allow a higher level node to be locked in S or X mode without having to check all descendent nodes.

In Table 3 , column 1 represents the lock type. Column 2 represents the value to be used in the input layer of the ANN in module 1 and module 3. Column 3 gives binary representation of Lock type to be used in the output layer of module 1 and module 3. The values are used as target outputs in the module 1 and module 3 during lock release on a data item.

Table 4 shows two transactions T1 and T2 in the first column. Each transaction requests object a or b with a lock mode S or X. The fourth column indicates if any one of the lock is assigned for the object and otherwise ‘0’ if no lock is assigned to the object.

Lock type	(Input layer representation numerical value).	Binary representation in target layer of the ANN
S	1	001
X	2	010
IS	3	011
IX	4	100
Object Not locked	0	000

TABLE 3: Binary representation of lock type.

User / Intermediate transaction	Object (a)	Object (b)	Mode S,X ,IS,IX	Lock Enabled – (1) Otherwise (0)
T1	a	-	S	1

T2	a	-	S	1
T1	a	-	X	1
T1	-	b	S	1
T2	a	-	X	1
T1	-	b	X	1

TABLE 4: Sample sequence of object access by two users.

This work uses for modules of algorithm which work using BPA given in Table 1. The modules given in Table 5 gives their usage for learning and finding the lock states. OML(Object, Mode, Lock) and OL (Object Mode)

Module	**Name**	**Training / Testing**	**ANN Topology**
1	UML	Training (Figure 4)	2{user number and mode} x {no. of nodes in hidden layer} x 3{Lock value}
2	UML	Testing (Figure 5)	2{user number and mode} x (no. of nodes in hidden layer) x 3(Lock value)
3	XL	Training (Figure 6)	1{user} x 2 {no. of nodes in hidden layer} x 3{lock value}
4	XL	Testing (Figure 7)	1{user} x 2 {no. of nodes in hidden layer} x 3{lock value}
In the fourth column of this table, 3 values are given in the order, no. of nodes in the input layer, no. of nodes in the hidden layer which can be anything and no. of nodes in the output layer which is 3(fixed)			

TABLE 5: Modules used for learning the lock status of an object.

OML training

IL **HL** **OL**

(Object name, mode are presented) **(Lock value is trained)**

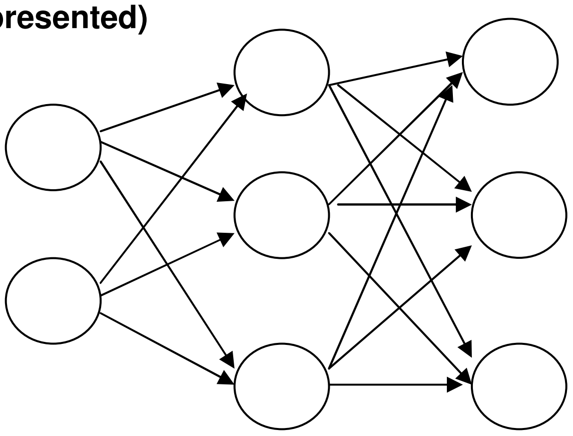

FIGURE 7: OML trining

OML testing

IL **HL** **OL**

(Object name, mode are presented) **(Lock value is obtained)**

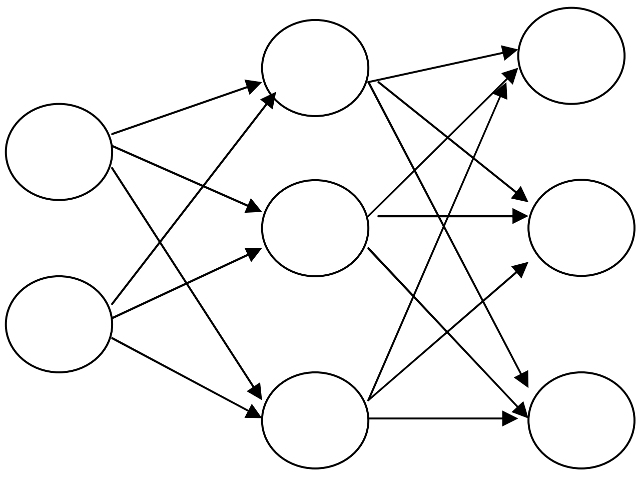

FIGURE 5: OML testing

OL training

IL **HL** **OL**

(Object name is presented) **(Lock value is trained)**

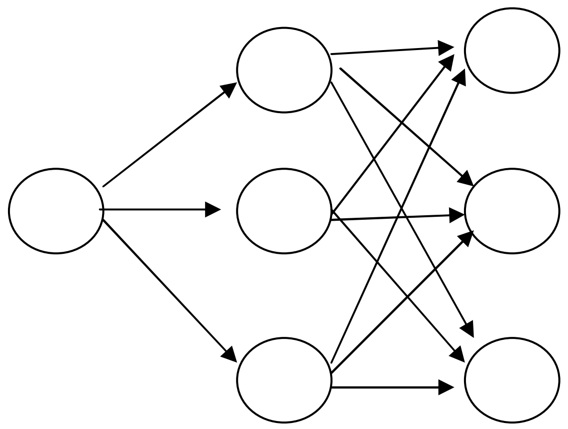

FIGURE 6: OL trianing

OL testing

IL **HL** **OL**

(Object name is presented) **(Lock value is obtained)**

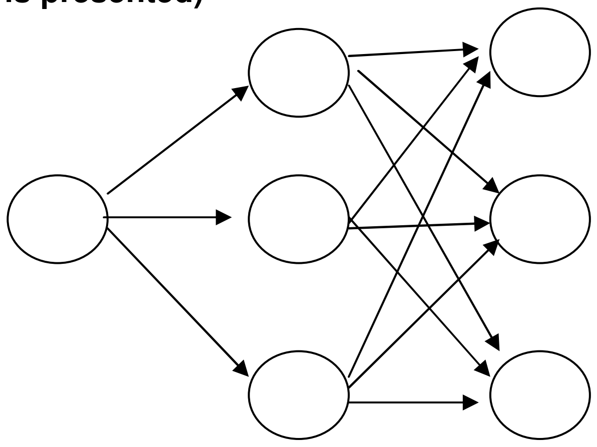

FIGURE 7: OL testing

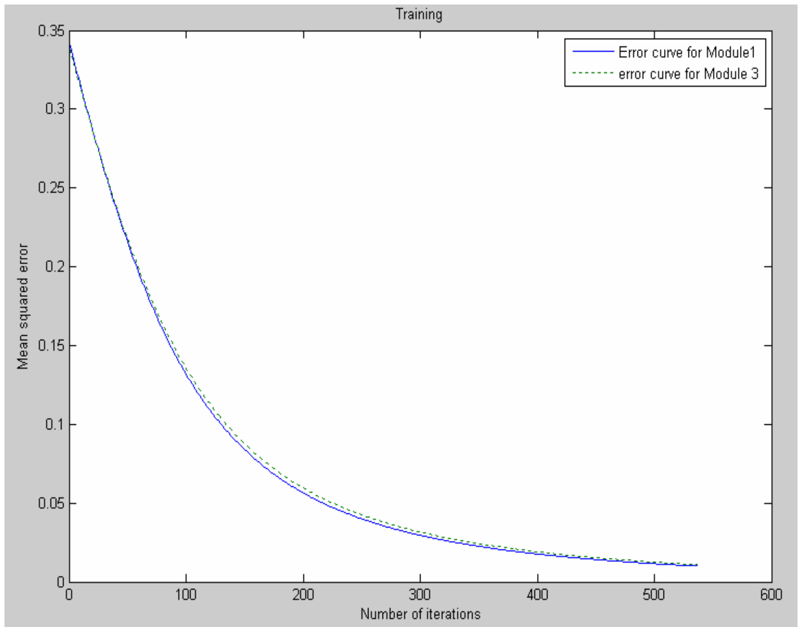

FIGURE 8: Mean squared error convergence.

Sequence of modules executed when a transaction requests lock or releases lock

1. Initialize randomly the weights of module 1 and module 3
2. A transaction T_i requests lock on an object (a ,b,)
3. Module 4 is tested with object (a,b, …) requested in step 2 to obtain binary value. If '000' is output in the output layer of module 4, then the object is free to be accessed. If (001, 010, 011, 100 is output then the object is under use. If the output value is 001, then the transaction in step is given access to the requested object
4. In any case , if T_i is given transaction to requested object, then module 1 and module 3 are weight updated using the backpropagation algorithm (forward and backward steps)
5. In any case , if the object is under any lock mode other than shared or no lock, then the transactions is kept under queue.

7. RESULTS AND DISCUSSIONS

The proposed ANN for lock state learning and lock state finding have been implemented using Matlab 7. Module 1 and Module 3 are trained until a Mean Square Error value of 0.01 is reached. The time for convergence to reach 0.01 is at an average of 3sec. The Figure 7, shows number of iterations versus mean squared error for Module 1 and Module 3.

8. CONCLUSION

An approach has been attempted to implement ANN in concurrency control. The approach has to be verified with different types of files operated by many users in a distributed environment. Different types of ANN algorithms can be attempted to achieve concurrency control in CAD database application.

9. REFERENCES

1. F. Bancilhon, W. Kim and H. Korth. “A Model of CAD Transactions”. In Proceedings of the 11th International Conference on Very Large Data Bases, Morgan Kaufmann, August, 1985, pp. 25-33

2. K. Salem, H. Garcia-Molina and R. Alonso. “Altruistic Locking: A Strategy for Coping with Long Lived Transactions”. In Proceedings of the 2nd International Workshop on High Performance Transaction Systems , September, 1987, pp. 19.1 - 19.24

3. P. Klahold, G. Schlageter, R. Unland and W. Wilkes, “A Transaction Model Supporting Complex Applications in Integrated Information Systems”. In Proceedings of the ACM SIGMOD International Conference on the Management of Data, ACM Press, May, 1985, pp. 388-401

4. A. H. Skarra, and S. B. Zdonik. “Concurrency Control and Object- Oriented Databases”. In Kim, W., and Lochovsky, F. H., Ed., *Object-Oriented Concepts, Databases, and Applications*, ACM Press, New York, NY, 1989, pp. 395-421

5. M. F. Fernandez and S. B. Zdonik. “Transaction Groups: A Model for Controlling Cooperative Work”. In Proceedings of the 3rd International Workshop on Persistent Object Systems, January, 1989

6. S. Purushothaman, Y. G. Srinivasa. “A back Propagation Algorithm applied to Tool wear Monitoring”. International Journal of Tools Manufacture ,1994, Vol 34, No 5, pp 625-631

7. S.Purushothaman, Y.G Srinivasa. “A procedure for training an artificial neural network with application to tool wear monitoring”. INT J.PROD RES., 1998, Vol., 36, No.3, 635-651

8. M. L. Brodie, B. Blanstein, U. Dayal, F. Manola, A. Rosenthal. “CAD/CAM Database Management”. IEEE Database Engineering, vo1.7, N0.2, pp. 12-20, June 1984

9. M. P. Groover and E. W. Zim mers. “CAD/CAM: Computer-Aided Design and Manufacturing”. Prentice-Hall, New York, *NY,* 1984

10. M. A. Ketabchi, V. Berzins. “Modeling and Managing CAD Databases”. IEEE Computer, February 1987, pp. 93-102

11. S. G. Landis. “Design Evolution and History in an Object-Oriented CAD/CAM Database”. IEEE, Proc.

12. Alexandtos Biliris, Huibin Zhao. “Design Versions in a Distributed CAD Environment”. 1989 IEEE, PP 354-359

13. A. A. Akintola, G. A. Aderounmu and A. U. Osakwe and M.O. Adigun. “Performance Modeling of an Enhanced Optimistic Locking Architecture for Concurrency Control in a Distributed Database System”. Journal of Research and Practice in Information Technology, Vol. 37, No. 4, November 2005