Implementation of Locally Weighted Projection Regression Network for Concurrency Control In Computer Aided Design

A.Muthukumaravel
Research Scholar,
Department of MCA,
Chennai – 600 117, India.

Dr.S.Purushothaman
Principal,
Sun College of Engineering
and Technology,
KK District – 629 902, India

Dr.A.Jothi
Dean,
School of Computing Sciences,
VELS University,
Chennai – 600 117, India.

***Abstract*—This paper presents implementation of locally weighted projection regression (LWPR) network method for concurrency control while developing dial of a fork using Autodesk inventor 2008. The LWPR learns the objects and the type of transactions to be done based on which node in the output layer of the network exceeds a threshold value. Learning stops once all the objects are exposed to LWPR. During testing performance, metrics are analyzed. We have attempted to use LWPR for storing lock information when multi users are working on computer Aided Design (CAD). The memory requirements of the proposed method are minimal in processing locks during transaction.**

Keywords-*Concurrency Control; locally weighted projection regression*; T*ransaction Locks; Time Stamping.*

I. INTRODUCTION

Concurrency control is one of the essential characteristics of transaction management to ensure consistency of database. Maintaining consistency in transactions of objects is mandatory. During computer aided design (CAD), many people will be accessing different parts of same objects according to the type work allotted to them. As all the parts of the same objects are stored in a single file, at any point of time, there should not be corruption of data, inconsistency in storage and total loss of data.

The concurrency control requires proper locking methods for controlled transactions. The most common way in which access to objects is controlled by 'locks'. In a database operation, lock manager plays an important role. It checks whether one or more transactions are reading or writing any object 'I' where 'I' is an item. It is the object of that record, for each item I. Gaining access to I is controlled by manager and ensure that there is no , access (read or write) would cause a conflict. The lock manager can store the current locks in a lock table which consists of records with fields (<object>, <lock type>, <transaction>) the meaning of record ('I', 'L', 'T') is that transaction 'T' has a lock of type 'L' on object 'I '[1-4].

The process of managing simultaneous operations on the database without having them interfere with one another is called concurrency. [5-8] When two or more users are accessing database simultaneously, concurrency prevents interference. Interleaving of operations may produce an incorrect result even though two transactions may be correct. Some of the problems that result in concurrency [11-20] are lost update, inconsistent analysis and uncommitted dependency.

II. PROBLEM DEFINITION

There is inability to provide consistency in the database when long transactions are involved. It will not be able to identify if there is any violation of database consistency during the time of commitment. It is not possible to know, if the transaction is with undefined time limit. There is no serializability when many users' work on shared objects. During long transactions, optimistic transactions and two phase locking will result in deadlock. Two phase locking forces to lock resources for long time even after they have finished using them. Other transactions that need to access the same resources are blocked. The problem in optimistic mechanism with 'Time Stamping' is that it causes repeated rollback of transactions when the rate of conflicts increases significantly. Artificial neural network [9] with locally weighted projection regression (LWPR) has been used to manage the locks allotted to objects and locks are claimed appropriately to be allotted for other objects during subsequent transactions.

Inbuilt library drawing for the dial of fork (Figure 1) is available in AutoCAD product. The fork is used in the front structure of a two wheeler. Due to customer requirements, the designer edits the dial of fork in the central database by modifying different features. Consistency of the data has to be maintained during the process of modifications of different features. A specific sequence of locking objects has to be done whenever a particular user accesses a specific feature of the dial of fork. Each feature is treated as an object. The features are identified with numbers. In this work, O1 refers to an object / feature marked as 1. The major objects involved in creating the dial of the fork are hollow cylinder, wedge and swiveling plate. The various constraints that have to be imposed during modifications of features by many users on this dial of fork are as follows:

- During development of features, hollow cylinder details should not be changed.

- External rings are associated with hollow cylinder.
- The circular wedge has specific slope and associated with hollow cylinder.

This dial of fork has following entities.

1) Features 1, 2, (set 1)
2) Features 10, 11,12,13,14 (set 2)
3) Features 5,6,7,8 (set 3)
4) Features 3, 4 (set 4)

Set 1, set 2, set 3, set 4 can be made into individual drawing part files (part file 1, part file 2, part file 3 and part file 4) and combined into one assembly file (containing the part files 1,2, 3 and 4 which will be intact). When the users are accessing individual part files, then transactions in part file 1 need not worry about the type of transactions in part files 2,3,4 and vice versa among them. When the part files 1, 2, 3 and 4 are combined into a single assembly file, then inconsistency in the shape and dimension of the set 1, set 2, set 3 and set 4, during matching should not occur. Provisions can be made in controlling the dimensions and shapes with upper and lower limits confirming to standards. At any time when a subsequent user is trying to access locked features, he can modify the features on his system and store as an additional modified copy of the features with Time Stamping and version names (allotted by the user / allotted by the system).

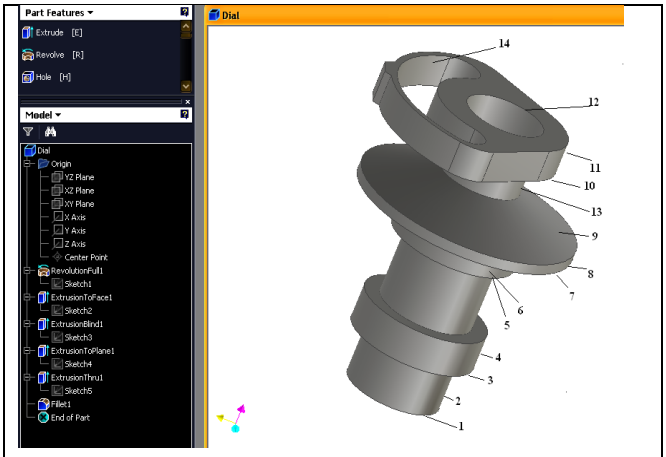

Figure 1. Dial of fork

1 Lower end, 2 Height of the end part, 3 External support, 4 Height of the external support, 5 Support for the wedge, 6.Height of the support for the wedge, 7 Wedge, 8.Thickness of the wedge, 9 Slope of the wedge, 10.Wedge lock, 11.Height of the wedge lock, 12 Concentric hole, 13. Separator, 14.Guideway

III. Locally Weighted Projection Regression

Locally Weighted Projection Regression (LWPR) [10] is an algorithm that achieves nonlinear function approximation in high dimensional spaces even in the presence of redundant and irrelevant input dimensions. At its core, it uses locally linear models, spanned by a small number of univariate regressions in selected directions in input space. The nonparametric local learning system

i. learns information rapidly with second order learning methods based on incremental training.

ii. uses statistically sound stochastic cross validation to learn information.

iii. adjusts its weighting kernels based on local information only.

iv. has a computational complexity that is linear in the number of inputs, and

v. can deal with a large number of possibly redundant and irrelevant – inputs.

The structure of the event loop is shown in Fig.2. The algorithm is at one of the four action states at any given point of time. The INITIALIZE phase is used to initialize the LWPR and read in the script variables from the script file and fill in default values for those variables not specified in the script file.

The TRAIN phase of the algorithm draws data from the training data set file and trains the local model on it. After every 'evaluation', the program goes into the EVALUATE phase where the learned model is tested against the novel (test) data set.

When the number of iterations has exceeded the 'max_iterations' count or the change of normalized mean squared error (nMSE) between the last EVALUATE phase and the current, a checking is done to find if the values is below a THRESHOLD. In such case, the program goes into the RESULT phase during which, it saves the learned LWPR.

A. *Adding an extra projection dimension*

The program can be initialized with 'init_n_reg=1'. It is taken as a special case of one projection LWPR where the number of local projection employed by each local model is restricted to one. The distance metric adjusts in order to accommodate for this restrictions. This may take a long time to converge for regression problems in which the inherent dimensionality is high. If there is no prior knowledge of the intrinsic local dimensionality, then initialize the variable 'init_n_reg=2'. This implies one dimension for the regression and the additional dimension used to check whether to add an extra projection or not. The mean squared error of the current regression dimension is compared against the previous one. Only if the ratio $\frac{nMSEr-1}{nMSEr} < \text{add\_threshold}$, a new regression is added. This criterion is used in conjunction with a few more checks to ensure that the decision is based on enough data support before deciding to add a new dimension.

B. *Pruning an existing RF (local model)*

If there is a local model that elicits substantial activation in response to a training data, it prevents the allocation of an additional local model for that training data. Since the distance metric is changing with the gradient descent updates, there can arise cases in which there is a considerable overlap between two local models. Due to this pruning of the variables required. The pruning is due to

i. too much overlap with another one (If 2 receptive fields elicits response greater than 'w_prune' to a training data), then one with the larger error is pruned.

ii. too high variance in the error compared to the 'std' of all the random field(RFs) (determined by the variable 'factor_prune').
iii. excessive error in one of the RFs.

C. Maintaining the local nearest neighbor (nn) list

When using the regression analysis in applications where the input values change smoothly, it is useful to keep a neighborhood list and perform training by looking at only the neighboring local models which are close to each other or have a substantial overlap in their activation profiles. This saves a lot of computing resources as opposed to going through all the local models and finding out those that have enough activation to be updated. It suffices to look at the neighborhood list to check for activations that are above the threshold and need to be updated.

D. Second order updates

The gradient descent updates of the distance metric is speeded up for faster convergence - and is more efficient if Newton's second order gradient information (meta learning) is used. If the 'allow_meta_learning' variable is TRUE, then the second order learning is on.

E. Forgetting factor

The forgetting factor is a variable that is used to discount the effects of the statistics computed at an earlier stage and give more weight to the recent statistics - which are a result of having experienced more data points. It can be thought of as a sliding window over which the stochastic sufficient statistics are accumulated. The forgetting factor ('lambda') takes a value [0, 1] where 0 corresponds to using only the current point and 1 corresponding to not `forgetting 'anything. An annealed forgetting rate can be used which forgets more at the start specified by 'init_lambda' and anneals towards a value closer to one ('final_ lambda') - not forgetting anything based on annealing factor 'tau_lambda'.

IV. Results And Discussion

Let us assume that there are two users editing the dial of the fork. User1 edits O1 and hence O2 will be locked sequentially (Table 1). Immediately user2 wants to edit O2, however he will not get transaction as already O2 is locked. However, user2 or any other user can try to access O3 to O14. The variables used for training the ANN about locks assigned to different objects are transaction id, object id, lock mode (Table 2).

TABLE I. Shape and dimension consistency management

Group	*First feature*	*Remaining feature to be locked*
G1	1	2
G2	10	11,12,13,14
G3	5	6,7,8
G4	3	4

Transaction id represents the client or any other intermediate transactions. Object id represents the entire feature or an entity in the file. Mode represents type of lock assigned to an object.

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

TABLE II. Binary representation of lock type

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

Initially, user 1 and user 2 have opened the same dial of fork file from the common database. The following steps shows sequence of execution and results

T1 edits O1 with write mode. Table 4 shows pattern formed for the training.

TABLE III. First time pattern used for training LWPR

Object number	*Input pattern*	*Target output pattern*
O_1	[1 1]	[0 1 0]

Step 1: The transaction manager locks objects mentioned in the third column of Table 1. Repeat step 1 with the patterns given in Table 4.

TABLE IV. Additional patterns used for training OML LWPR

Object number	*Input pattern*	*Target output pattern*
O_1	[1 1]	[0 1 0]
O_2	[2 1]	[0 1 0]

Step 2: A new transaction T2 access O2. A pattern is formed to verify if lock has been assigned to O2 and its associated objects O1. Only when the locks are not assigned to O2 and O1 then T2 is allowed.

The following input patterns are presented to the testing

module to find if the output [0 0 0] is obtained in the output layer. During testing, the final weights obtained during training will be used. Otherwise it means that lock has been assigned to either O2. In such case, transaction is denied for T2. Else the following Table 5 is presented in step 1.

TABLE V. ADDITIONAL PATTERNS USED FOR TRAINING LWPR

Object number	*Input pattern*	*Target output pattern*
O_1	[1 1]	[0 1 0]
O_2	[2 1]	[0 1 0]
O_3	[3 1]	[0 1 0]
O_4	[4 1]	[0 1 0]
O_5	[5 1]	[0 1 0]
O_6	[6 1]	[0 1 0]
O_7	[7 1]	[0 1 0]
O_8	[8 1]	[0 1 0]

Step 3: To know the type of lock value assigned to an object and for a transaction, testing is used. Testing uses the final weights created by training. The proposed LWPR for lock state learning and lock state finding have been implemented using Matlab 7.

The performance of the LWPR algorithm has been presented on the following criteria.

1. Locking time for each object. (Figure 3)
2. Releasing time for each object.(Figure 4)
3. Total Locking time for each transaction group.(Figure 5)
4. Arrival rate (Figure 6)
5. Response time (Figure 7)

V. CONCLUSION

An artificial neural network with LWPR has been implemented for providing concurrency control to maintain consistency in the CAD database. A dial of fork has been considered that contains 14 objects. The 14 objects have categorized into 4 groups. The transaction behavior and concurrency control by the two users on the 14 objects have been controlled using LWPR network. The LWPR method requires less memory based on the topology used for storing objects and its transactions when compared with conventional method.

REFERENCES

[1] Rosenkrantz .D , Stearns R .and. Lewis P, "System-level concurrency control for distributed database systems," In ACM Transactions on Database Systems, vol. 3, No. 2, . 1978, pp. 178-198,

[2] Peter A. Buhr, Ashif S. Harji, Philipp E. Lim and Jiongxiong Chen,'Object-oriented real-time concurrency', In Proceedings of the 15th ACM SIGPLAN conference on Object-oriented programming systems, languages and applications, 2000,pp. 29-46.

[3] Mihalis Yannakakis, 'Issues of correctness in database concurrency control by locking', In Proceedings of the thirteenth annual ACM symposium on Theory of computing, 1981,pp. 363–367.

[4] Klahold .P , Schlageter G. and Wilkes W., August, 'A General Model for Version Management in Databases', In Proceedings of the International Conference on Very Large Data Bases, Kyoto, 1986,pp. 319-327.

[5] Katz R.H. and Lehman T.J, 'Database Support for Versions and Alternatives of Large Design Files', In IEEE Transactions on Software Engineering, vol. 10, No. 2, ., March 1984,pp. 191-200.

[6] Herrmann U., Dadam P., Küspert K., Roman E. A. and Schlageter G., 1990, 'A lock technique for disjoint and non-disjoint complex objects', In Springer Advances in Database Technology — EDBT '90, vol. 416, pp. 219-237.

[7] Garza J. and Kim W., 'Transaction management in an object-oriented database system'. In Proceedings of the ACM SIGMOD International Conference on the Management, Vol. 17,No. 3,1988,,pp. 37-45.

[8] Eliot B. Moss, , 'Transaction Management for Object-Oriented Systems',In Proceedings of the IEEE Computer Society International Workshop on Object-Oriented Database Systems, 1986,pp. 229.

[9] Raviram P., Wahidabanu R. S. D. and Purushothaman S., **"**Concurrency Control in CAD with KBMS using Counter Propagation Neural Network**",** IEEE International Advance Computing Conference, 6-7 March 2009, pp. 1521-1525.

[10] Purushothaman S., Elango M.K. and Nirmal Kumar S., Application of Hilbert Huang Transform with Locally Weighted Projection Regression Method for Power Quality Problems, International Review on Electrical Engineering, vol. 5. no. 5, October 2010, pp. 2405-2412

[11] Mohammed Khaja Nizamuddin, Dr. Syed Abdul Sattar, 2010, "Data Count Driven Concurrency Control Scheme with Performance Gain in Mobile Environments" in Journal of Emerging Trends in Computing and Information Sciences, vol 2 ,no. 2, pp 106-112.

[12] Salman Abdul Moiz, Dr. Lakshmi Rajamani, "An Algorithmic approach for achieving Concurrency in Mobile Environment", INDIACom, 2007,pp.209-211,.

[13] K M Prakash Lingam, 2010, "Analysis of Real-Time Multi version Concurrency Control Algorithms using Serialisability Graphs" International Journal of Computer Applications (0975 - 8887), vol 1 ,no. 21, pp. 57 – 62.

[14] Quilong Han,Haiwei pan, "A Concurrency Control Algorithm Access to Temporal Data in Real-Time Database Systems," imsccs, , International Multi symposiums on Computer and Computational Sciences,2008, pp.168-171.

[15] Tae-Young Choe, 2008, "Optimistic Concurrency Control based on Cache Coherency in Distributed Database Systems" IJCSNS International Journal of Computer Science and Network Security, vol.8, no.11, November 2008, pp 148–154

[16] Mohammed Khaja Nizamuddin, Syed Abdul Sattar, "Adaptive Valid Period Based Concurrency Control Without Locking in Mobile Environments" In: Recent Trends in Networks and Communications, Springer CCIS, vol.90, Part 2, , 2010, Springer Heidelberg, pp 349-358

[17] Arun Kumar Yadav & Ajay Agarwal, "An Approach for Concurrency Control in Distributed Database System" in International Journal of Computer Science &Communication, vol. 1, no. 1, January-June 2010, pp. 137-141

[18] Arumugam.G and Thangara.M "An Efficient Locking Model For Concurrency Control In Oodbs" Data Science Journal, vol 4, 31 August 2005, pp 59-66.

[19] Poonam Singh, Parul Yadav, Amal Shukla and Sanchit Lohia "An Extended Three Phase Commit Protocol for Concurrency Control in Distributed Systems" in International Journal of Computer Applications (0975 – 8887) vol 21,no.10, May 2011

[20] Jinhua Guo, "An Exploratory Environment for Concurrency Control Algorithms" in International Journal of Computer Science vol 1 ,no 3, March 29, 2006, pp 203-211.

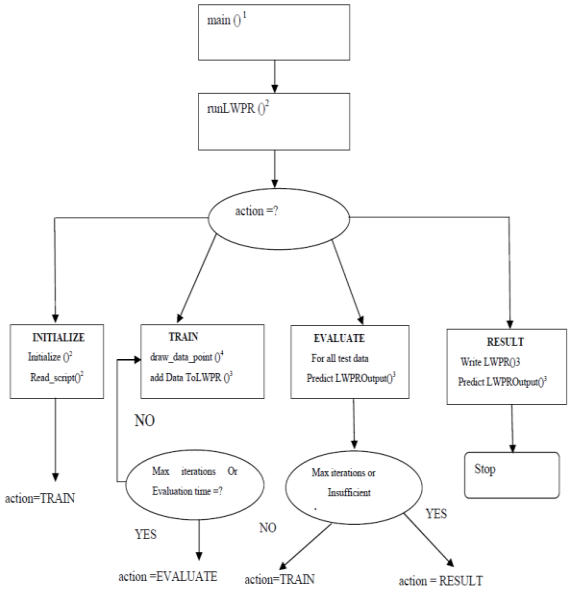

Figure 2. LWPR training modules

Filenames: 1.lwpr_main(), 2.lwpr_test(), 3.lwpr(), 4.utilities()

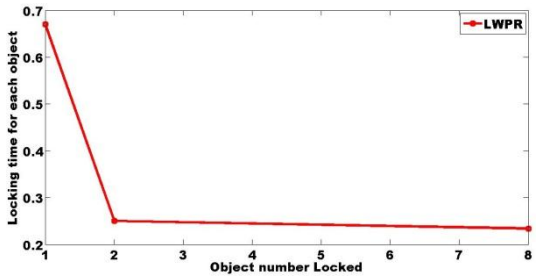

Figure 3. Locking time for each object

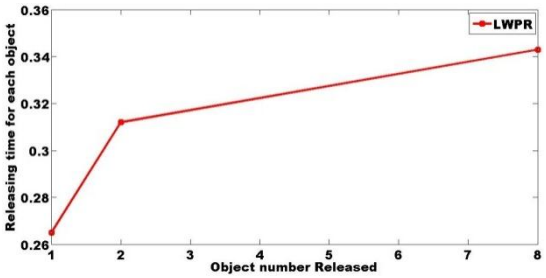

Figure 4. Releasing time for each object

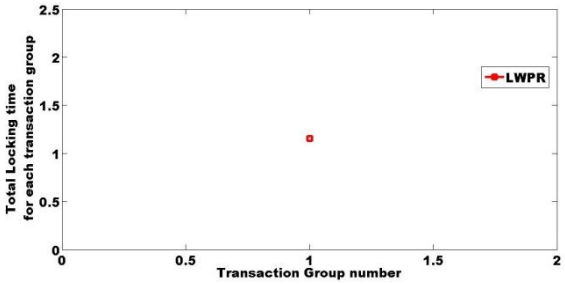

Figure 5. Locking time for each group

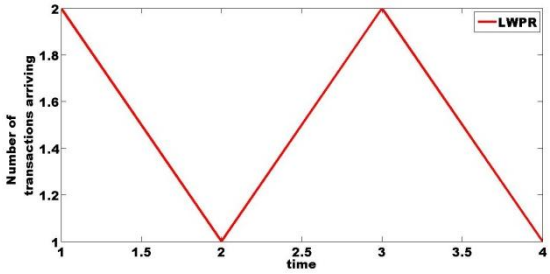

Figure 6. Arrivals rate

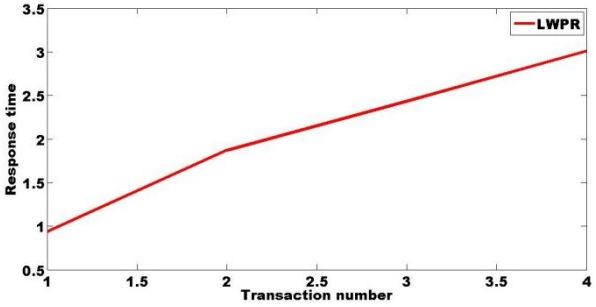

Figure 7. Response time