Development of Locally Weighted Projection Regression for Concurrency Control in Computer-Aided Design Database

Purushothaman Srinivasan
Adama Science and Technology University
88 PUBLICATIONS **82** CITATIONS

Development of Locally Weighted Projection Regression for Concurrency Control in Computer-Aided Design Database

A. Muthukumaravel, S. Purushothaman and R. Rajeswari

Abstract Concurrency control (CC) is the activity of synchronizing operations issued by concurrently executing programs on a shared database. Concurrency control is an important concept for proper transactions on objects to avoid any loss of data or to ensure proper updating of data in the database. This paper presents development of locally weighted projection regression (LWPR) for concurrency control while developing bolted connection using Autodesk inventor 2008. While implementing concurrency control, this work ensures that associated parts cannot be accessed by more than one person due to locking. 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. We have attempted to use LWPR for storing lock information when multi users are working on computer-aided design (CAD).

Keywords Concurrency control · Locally weighted projection regression · Transaction locks · Time stamping

A. Muthukumaravel (✉)
Department of MCA, School of Computing Sciences, VELS University,
Chennai 600 117, India
e-mail: muthu14673@gmail.com

S. Purushothaman
PET Engineering College, Tirunelveli District, Tamil Nadu 627 117, India
e-mail: drsppuru@gmail.com

R. Rajeswari
Department of Computer Science, Mother Teresa Womens University,
Kodaikanal, India
e-mail: rajeswaripuru@gmail.com

Introduction

Concurrency control is a set of mechanisms used to maintain consistency in transactions of databases. Proper locking method for controlled transactions. The most common way in which access to items is controlled by “locks-L”. Lock manager is the part of a database management system (DBMS) those records, for each item ‘O’, whether one or more transactions are reading or writing any part of ‘O’. If so, the manager will forbid another transaction from gaining access to ‘O’, provided the type of access (read or write) could cause a conflict. The lock manager can store the current locks in a lock table which consists of records with fields (< object-‘O’ > , < lock type-‘L’ > , < transaction-‘T’ >) the meaning of record (‘O’, ‘L’, ‘T’) is that transaction ‘T’ has a lock of type ‘L’ on object ‘O’ [1-3].

Concurrency problems in data base systems arise from controlling concurrent updates on a shared data base so that (1) no lost updates can occur, (2) data base consistency can be maintained, and (3) noncascade backup is possible.

The advantages of two phase locking (2PL) and time stamping have been taken as the initial starting point and improvements have been achieved by implementing LWPR for locking objects to achieve concurrency control in long time transactions of CAD/CAM database system.

The concurrency control requires proper locking method for controlled transactions. The most common way in which access to items is controlled by “locks”. Locks are used for accessing objects [4-8].

In this research work, LWPR has been implemented. The performances of the algorithm have been compared based on the following criteria.

1. Locking time for each object
2. Releasing time for each object
3. Total Locking time for each transaction group
4. Total Releasing time for each transaction group.

Locally Weighted Projection Regression

LWPR is an algorithm that achieves nonlinear function approximation in high dimensional spaces even in the presence of redundant and irrelevant input dimensions [3, 14, 15]. At its core, it uses locally linear models, spanned by a small number of univariate regressions in selected directions in input space. This nonparametric local learning system

1. learns rapidly with second order learning methods based on incremental training.
2. Uses statistically sound stochastic cross validation to learn.

3. Adjusts its weighting kernels based on local information only.
4. Has a computational complexity that is linear in the number of inputs.
5. Can deal with a large number of—possibly redundant and irrelevant—inputs.

The structure of the event loop is shown in Fig. 1. 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.

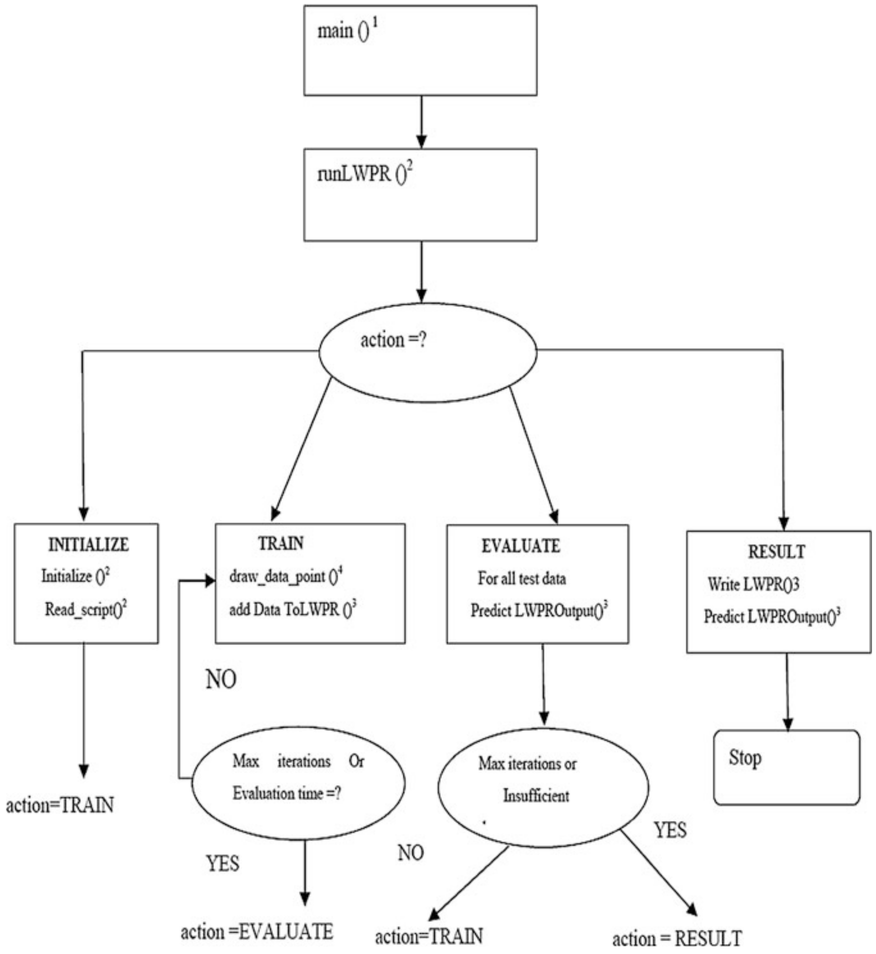


Fig. 1 LWPR training modules Filenames: 1.lwpr_main(), 2.lwpr_test(), 3.lwpr(), 4.utilities()

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.

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 [9, 10].

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 [11]-[13]. 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.

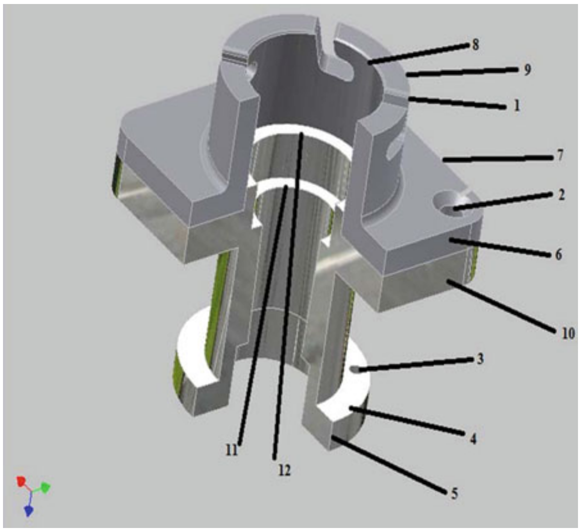
A bearing-type connection is the most common type of bolted connection. It is used in most simple-shear connections and in situations when loosening or fatigue due to vibration. A slip-critical connection is one in which loosening due to vibration or load reversals are to be considered. Bolted connection are easily disassembled. They can be designed to take tension loads (Fig. 2).

Inbuilt library drawing for the bolted connection are available in Autodesk inventor 9. Drawings considered for this research work are:

In general, the following sequences are formed when creating bolted connection. Even though library files are available for bolted connection drawing, customized drawing bolted connection file is discussed [16]-[19]. The major parameters involved in creating the bolted connection are diameter (outer diameter/inner diameter), depth (hole)/length (extension or extrusion or projection) and chamfering (slanting). The various constraints that have to be imposed during modifications of features by many transactions on this bolted connection are.

1. During chamfering, both length and diameter have to be write locked at the first transaction T1.
2. During diameter modification, chamfering has to be locked.
3. During length or depth modification, diameter has to be locked.

Fig. 2 Bolted connection *1* Locking slot – 4 numbers, *2* Bolt Fixing – 1, *3* Bolt Fixing – 2, *4* Lower base support width of annular, *5* Lower base support height, *6* Height of bolt fixing, 7 Square upper late, *8* Inner diameter, *9* Outer diameter, *10* Square lower plate, *11* Concentric 1, *12* Concentric 2



4. This bolted connection has two major entities.
1. Features 1-2-10-11-12 (set 1)
2. Features 3-4 (set 2)
3. Features 5-6-7-8-9 (set 3)

Set 1, set 2 and set 3 can be made into individual drawing part files (part file 1 part file 2 and part file 3) and combined into one assembly file (containing the part files 1, 2, and 3 are intact). When the transactions are accessing individual part files, then transactions in part file 1 about the type of transactions in part files 2, 3 and vice versa. However, when the part files 1, 2 and 3 are combined into a single assembly file, then inconsistency in the shape and dimension of the set 1, set 2 and set 3, during matching should not occur. Hence, provisions can be made in controlling the dimensions and Shapes with upper and lower limits confirming to standards. At any part of time when a subsequent transaction is trying to access locked features, one can modify the features on the system and store as an additional modified copy of the features with time stamping and version names (allotted by the transaction/allotted by the system).

Let us assume that there are two transactions editing the bolted connection. Transaction 1 edits O_1 and hence O_8 and O_9 will be written locked sequentially (Table 1). Immediately transaction 2 wants to edit O_2 and hence O_6, O_7 will be locked in sequence. Any other transaction can access O_{10} or O_4 or O_3.

Initially, transaction 1 and transaction 2 have opened the same bolted connection file from the common database. The following steps shows sequence of execution.

Step 1: For the first time, a pattern is generated as soon as a transaction accesses the object. T_1 access O_1 with write mode. Table 2 shows pattern formed for the OL training.

Table 1 Bolted connection shape and dimension consistency management

Group	First feature	Remaining feature to be locked
G1	1	8,9
G2	2	6,7
G3	10	12
G4	4	5,11
G5	3	4

Table 2 T_1 First pattern used for training OL

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

Table 3 T_1 Additional patterns used for training OL

Object number	Input pattern	Target output pattern
O_1	[1]	[0 1 0]
O_8	[8]	[0 1 0]
O_9	[9]	[0 1 0]

Table 4 T_2 Additional patterns used for training OL

Object number	Input pattern	Target output pattern
O_1	[1]	[0 1 0]
O_8	[8]	[0 1 0]
O_9	[9]	[0 1 0]
O_2	[2]	[0 1 0]

Step 2: The transaction manager locks objects mentioned in the third column of Table 1 Now, repeat step 1 with the patterns given in Table 3.

Step 3: A new transaction T_2 accesses O_2. A pattern is formed to verify if lock has been assigned to $O_{2\ \text{and}}$ its associated objects O_6, O_7. Only when the locks are not assigned to O_6 and O_7 then T_2 is allowed.

The patterns of additional transactions are presented to the OL testing module to find if the output [0 0 0] is obtained in the output layer. During OL testing, the final weights obtained during OL training will be used. Otherwise, it means that lock has been assigned to either O_2 or O_6 or both. In such case, transaction is denied for $T_{2.}$ Else the following patterns in Table 4 are presented in step 1.

Step 4: To know the type of lock value assigned to an object and for a transaction, OL testing is used. OL testing uses the final weights created by OL training.

Results

Figures 3, 4, 5, and 6 shows the preformance of CC during locking and releasing of bolted connection object. For bolted connection object the performances have been evaluated using LWPR under controlled environment.

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

1. Releasing time for each object in bolted connection (Fig. 3).
2. Locking time for each object in bolted connection (Fig. 4).
3. Total Locking time for each transaction group in bolted connection (Fig. 5).
4. Total Releasing time for each transaction group in bolted connection (Fig. 6).

Bolted connection drawing

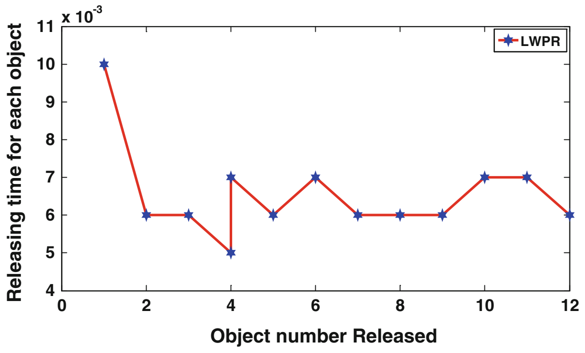


Fig. 3 Releasing time for each object in bolted connection

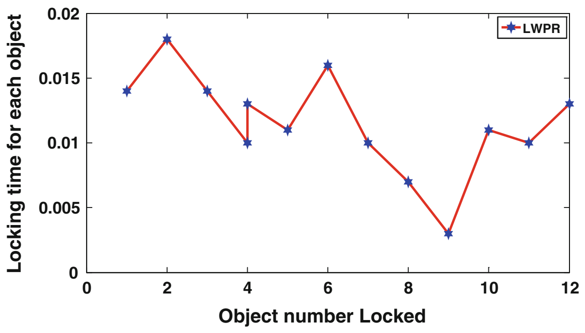


Fig. 4 Locking time for each object in bolted connection

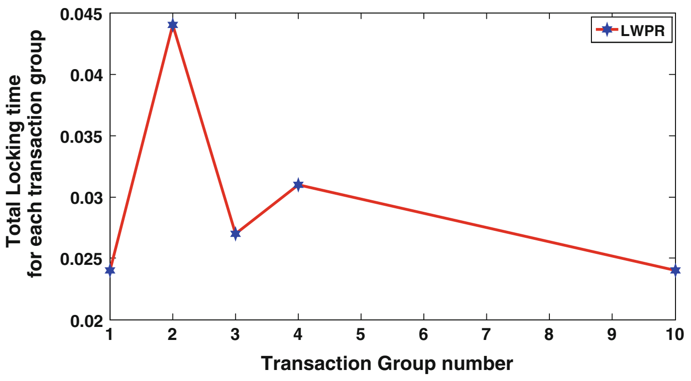


Fig. 5 Total locking time for each transaction group in bolted connection

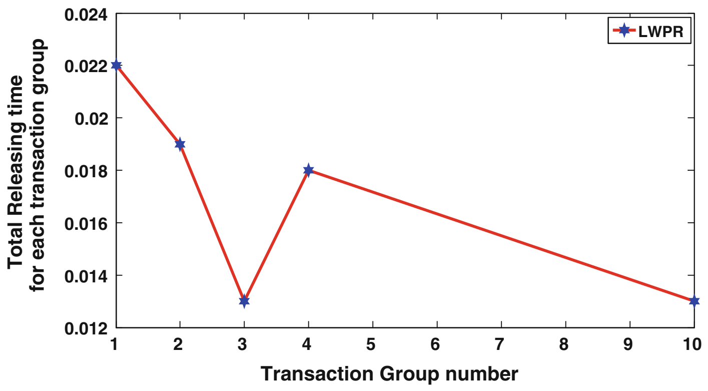


Fig. 6 Total releasing time for each transaction group in bolted connection

Conclusion

A LWPR has been implemented for providing concurrency control to maintain consistency in the CAD database. The performance of the LWPR algorithm is based on the Locking time and releasing time for each object, Total Locking time for each transaction group, Total Releasing time for each transaction group and Computation complexity. The LWPR algorithm takes less number of iterations to reach a stable state.

References

1. Buhr, P.A., Harji, A.S., Lim, P.E., Chen, J.: Object-oriented real-time concurrency. In: Proceedings of the 15th ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages and Applications, pp. 29–46, ACM New York, NY, USA (2000)
2. Raviram, P., Wahidabanu, R.S.D., Purushothaman, S.: Concurrency control in CAD with KBMS using counter propagation neural network. IEEE International Advance Computing Conference, pp. 1521–1525, 6–7 March, Patiala (2009)
3. Purushothaman, S., Elango, M.K., Nirmal Kumar, S.: Application of Hilbert Huang transform with locally weighted projection regression method for power quality problems. Int. Rev. Elect. Eng. **5**(5) 2405–2412 (2010)
4. Nizamuddin, M.K., Sattar, S.A.: Data count driven concurrency control scheme with performance gain in mobile environments. J. Emerg. Trend. Comput. Inform. Sci. 2(2), 106–112 (2010)
5. Moiz, S.A., Rajamani, L.: An algorithmic approach for achieving concurrency in mobile environment. In: INDIACom, pp. 209–211, (2007)
6. Lingam, K.M.P.: Analysis of real-time multi version concurrency control algorithms using serialisability graphs. Int. J. Comp. Appl. (0975 - 8887) **1**(21), 57–62 (2010)
7. Han, Q., Pan, H.: A concurrency control algorithm access to temporal data in real-time database systems. In: International multi symposiums on computer and computational sciences, pp. 168–171, IEEE Computer Society Washington, DC, USA (2008)
8. Choe, T.-Y.: Optimistic concurrency control based on cache coherency in distributed database systems. Int. J. Comp. Sci. Netw. Secur. 8(11), 148–154 (2008)
9. Nizamuddin, M.K., Sattar, S.A.: Adaptive valid period based concurrency control without locking in mobile environments. In: Recent Trends in Networks and Communications, vol. 90, part 2, pp. 349–358. Springer CCIS, Springer, Heidelberg (2010)
10. Yadav, A.V., Agarwal, A.: An approach for concurrency control in distributed database system. Int. J. Comp. Sci. Commun. 1(1), 137–141 (2010)
11. Arumugam, G., Thangara, M.: An efficient locking model for concurrency control in OODBS. Data Sci. J. **4**(31), 59–66 (2005)
12. Singh, P., Yadav, P., Shukla, A., Lohia, S.: An extended three phase commit protocol for concurrency control in distributed systems. Int. J. Comp. Appl. **21**(10), 0975–8887 (2011)
13. Guo, J.: An exploratory environment for concurrency control algorithms. Int. J. Comp. Sci. **1**(3), 203–211 (2006)
14. Vijayakumar, S., Schaal, S.: Locally weighted projection regression: an O(n) algorithm for incremental real time learning in high dimensional spaces. In: Proceedings ICML, vol. 1, pp. 288–293, Los Angeles, USA (2000).
15. Vijayakumar, S., Schaal, S.: Locally weighted projection regression: an *O(n)* algorithm for incremental real time learning in high dimensional space. In: Proceedings of 17th International Conference on Machine Learning, pp. 1079–1086, Los Angeles, USA (2000)
16. Ghosh, S.K., Islam, M.S., Lee, S.-Y., Liou, R.-L.: A multi-granularity locking model for concurrency control in object – oriented database systems. IEEE Trans. Knowl. Data Eng. 8(1), 144–155, (1996)
17. Kuo, T.-W., Wu, J., Hsih, H.-C.: Real-time concurrency control in a multiprocessor environment. IEEE Trans. Parallel Distrib. Syst. **13**(6), 659–671 (2002)
18. Park, S.-K.: Seismic performance test of bolted connections between precast-concrete column and H-beam. In: 8th Russian-Korean International Symposium on Science and Technology 2004 Proceedings, vol. 2, pp. 335–339, Ulsan University, South Korea
19. Rahman, M.A.: On analytical performance measurement of concurrency control protocols in DBMS. Int. J. Comp. Elect. Eng. **1**(3), 284–287 (2009)