Evolutionary Neural Network Learning Algorithms for Changing Environments*

MIGUEL ROCHA[1], PAULO CORTEZ[2] and JOSÉ NEVES[1]
[1]Departamento de Informática / [2]Dep. Sistemas de Informação
Universidade do Minho
[1]4710-057 Braga / [2]4800-058 Guimarães
PORTUGAL
mrocha@di.uminho.pt pcortez@dsi.uminho.pt jneves@di.uminho.pt
http://www.di.uminho.pt/~pcortez/ http://www.dsi.uminho.pt/~jneves/

Abstract: Classical *Machine Learning* methods are usually developed to work in static data sets. Yet, real world data typically changes over time and there is the need to develop novel adaptive learning algorithms. In this work, a number of algorithms, combining *Neural Network* learning models and *Evolutionary Computation* optimization techniques, are compared, being held several simulations based on artificial and real world problems. The results favor the combination of evolution and lifetime learning according to the *Baldwin effect* framework.

Key-Words: Baldwinian and Lamarckian Effects, Evolutionary Programming, Multilayer Perceptrons.

1 Introduction

In recent years, there has been a remarkable development of the *Data Mining* and *Machine Learning (ML)* arenas, which aim at the extraction of patterns or models from observed data. This interest arose due to an ever-increasing load of data, which often presents high complexity, while human experts are limited and may overlook important details. Yet, the majority of these techniques use static *off-line* learning (where the data is stored and accessed repeatedly), while in many applications data is rather changing and evolving [5]. In these cases, there is a clear need for *adaptive learning*, where the learning function changes over time.

On the other hand, living creatures have survived in hazardous and dynamic environments for millions of years, by taking advantage of two interacting processes: *evolution* and *lifetime learning*. *Evolution* is a slow process that takes place at the population level, determining the basic structure of an organism, while *lifetime learning* is responsible for the adaptation at the individual's level. In computational terms, these can be materialized via the *Evolutionary* and *Neural Computation* fields. The former is suitable for global search and the latter for fine tuning (local search).

When combining evolution and learning, two major frameworks can be addressed, namely the *Lamarckian* [4] and *Baldwinian* strategies [1], which differ on the fact that acquired traits are recoded into the chromosome or not. Although there has been past work in this arena, most of the studies consider only a subset of the possible strategies or focus mostly on the benefits of lifetime learning, being the trade-off between costs and profits rarely considered [9].

In this work, the *evolution* process is approached via *Evolutionary Programming (EP)*, where each individual encodes the weights of a *MultiLayer Perceptron (MLP)* [8]. *Lifetime learning* is addressed by the training of the *MLP*, using a gradient descent based procedure [7]. Five different strategies will be tested in several classification tasks, being the comparisons based on computation time, so that they can be fairer.

2 Changing Environments

The experiments that will be considered in this work endorse an important *ML* category, *classification* tasks [5], where the aim is to label a specific data item to a categorical class. In this study, two artificial and two real world tasks were selected:

6 Bit Classification (6BC) - Set by 2^6 binary patterns of 6 inputs and 1 output, whose value is set to 1 if the number of true input bits is odd [7].

*This work was supported by the FCT project POSI/ROBO/43904/2002 (partially funded by FEDER).

```
BEGIN
   Select P initial examples (from the database)
   FOR ( each event at time t ∈ {t_1, ..., t_N} ) DO
      Discard E examples
      Select new E examples (from the database)
END
```

Fig. 1: Pseudo-code for the dynamic environments.

Three Color Cube (TCC) - The painting of a 3D cube made up by a 3x3x3 grid of blocks [4]. The inputs stand for the blocks' coordinates on the X, Y and Z axis and the output denotes a color (the corners are black, the cubes in the center are white, the other are gray).

Contraceptive Method Choice (CMC) - Indonesian contraceptive method adoption, based on a number of socio-economical attributes (e.g. *wife's age*) [3]. The data has 1473 instances with 9 inputs (2 continuous, 4 categorical and 3 binary) and 3 choices (*no use*, *long-term* or *short-term*).

Car Evaluation Database (CEB) - The evaluation of cars (4 labels) according to *price*, *technical* and *comfort* aspects, with a total of 1728 examples and 6 nominal inputs [3].

Although these tasks are static, is is possible to simulate changing environments using the dynamic process described in Figure 1, which requires the definition of the following parameters: P, the number of patterns to be fed to the learning algorithm in each period; E, the number of examples that are changed at each event; and $t_1, \ldots, t_N$, the set of time events that will induce changes. The last two factors will influence the type of environment: *soft changing* (or *concept drift*), when changes occur gradually; and *hard changing* (or *concept shift*), when changes occur abruptly. Both types of environments will be tested:

Soft (S) - one pattern will change at each second ($E = 1$, $t \in \{1s, 2s, 3s, \ldots\}$); and

Hard (H) - several patterns will be commuted over a wider period ($t_{i+1} - t_i = 20s$ for the artificial tasks and $50s$ for real ones).

Since the artificial tasks (*6BC* and *TCC*) present few examples, these were adapted according to:

$6BC_S$ - the first output patterns ($P = 64$) are randomly set; then, during each event, one output is selected to change (from 0 to 1 or the reverse);

Table 1: The *MLP* topologies.

Task	Inputs	Hidden	Outputs	Weights
6BP	6	6	1	49
TCC	3	8	3	59
CMC	21	10	3	253
CEB	21	10	4	264

$6BC_H$ - the *6BC* task, except that all the desired outputs ($E = P = 64$) will flip periodically;

TCC_S - each cube is initially assigned a random color, and then, at each event, the color of one randomly chosen cube is changed;

TCC_H - the static rules prevail, although the cubes are periodically repainted ($E = P = 27$), following a predefined order (black follows gray, gray follows white and white follows black).

For the tasks CMC and CEB, the parameters were set to E=1, P=300 (***S***); and E=50, P=300 (***H***).

3 Learning Algorithms

Multi-Layer Perceptrons (MLPs) are one of the most popular *Neural Networks*, where *neurons* are grouped in *layers* and only *forward connections* exist [2]. *MLPs* are appealing due to their capability to model complex multi-dimensional data, even when noise is present, often outperforming other *ML* techniques [6].

In this work, fixed fully connected *MLPs* were adopted, with one hidden layer, *bias* connections and *logistic* activation functions. Before learning, the input (output) values were rescaled to the range $[-1, 1]$ ($[0, 1]$), using a *1-of-C* encoding scheme for the nominal attributes (with more than two possible classes). The topologies were chosen to make the learning possible with a minimum complexity (Table 3).

Five distinct algorithms will be defined to approach each learning task:

The *Connectionist Model (CM)*. Under these circumstances, the learning is achieved by a single *MLP* and the training performed by the *RPROP* algorithm [7].

The *Population of Connectionist Models (PM)*. A set of 20 *MLP* individuals will improve only via the learning algorithm (*RPROP*); i.e., no reproduction or selection procedure is applied.

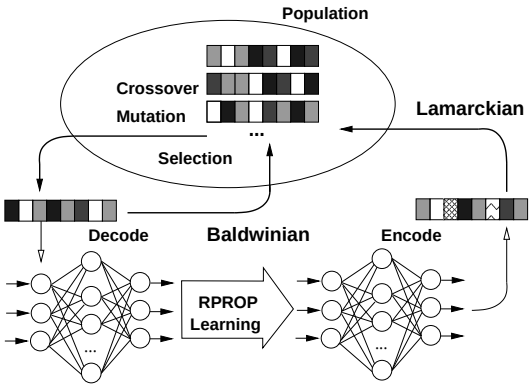

Fig. 2: The Baldwinian and Lamarckian strategies.

The *Darwinian Model (DM).* The learning process is accomplished by *EP*, where a population of 20 real-valued chromosomes is evolving, each coding the weights of a *MLP* [8]. In each iteration, 50% of the individuals are kept from the previous generation, being the remaining bred through the application of a *gaussian mutation*, adding a small perturbation to a variable number of genes (this value is randomly set between 1 and 20% of the total number of weights).

The *Lamarckian Model (LM).* The *LM* combines both *lifetime learning* and *evolutionary* approaches. *EP* is still used, although the mutation operator is replaced by a *random* one, which replaces a number of weights by values in the range $[-1.0, 1.0]$. In previous work, this macro-mutation strategy has achieved better results [8]. Each individual is allowed to learn during its lifetime, by running the *RPROP* algorithm for 50 epochs in each generation of the *EP* process. Then, the improved weights are encoded back into the chromosome (Figure 2).

The *Baldwinian Model (BM).* Similar to the *LM*, except that lifetime learning is only used to improve the fitness of the individuals, and the new weights are not encoded back into the genome (Figure 2). This means that, in the process of reproduction, the offspring does not inherit the acquired genetic information from their ancestors.

For all models, the initial weights are randomly set within the range $[-1.0; 1.0]$. The accuracy is measured by the *Root Mean Squared Error (RMSE)*:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{P}\sum_{j=1}^{O}(T_{i,j} - MLP_{i,j})^2}{PO}} \qquad (1)$$

where P denotes the number of the training patterns; O the number of the *MLP* outputs; $T_{i,j}$ the target and $MLP_{i,j}$ the network value; both for the output j and the i input pattern. This metric is used as the fitness value in the evolutionary approaches. For the population based models (*PM*, *DM*, *LM* and *BM*), the overall accuracy is given by the *RMSE* of the best individual.

4 Results

All experiments were conducted using *Java* programming environments, running on a *PC* with a *Pentium V 2.4 GHz* processor. The results were compared in terms of two parameters: the overall learning's *accuracy* (*RMSE*), and the process' *efficiency* (time elapsed, in seconds). For all models, at each time slot, one considers the average of the results obtained in 20 independent runs. Finally, the termination criteria was set by a time limit T, here set to $T = 1000s$.

Regarding the *soft* changing experiments (Figure 5), the *CM* is incapable of escaping the initial learning bias, presenting the worst performance. Both *PM* and *DM* show stable patterns, but seem unable to improve. When combining evolution and learning better results are achieved, being the *BM* the best alternative.

For the *hard* changing tasks, only the best strategies (*BM* and *LM*) were plotted, to simplify the visual analysis (Figure 6). Here, the error ranges are higher for the artificial tasks, which is explained by the larger number of commuted patterns. Again, and despite the more radical changes, the *BM* still manages to get a steady performance over time, being the best solution.

To achieve a global accuracy, the values within the first 200 seconds were discarded (a start-up time set to allow the tunning of each model); then, a *box plot* was computed over the remaining values, including the statistics (from bottom to top): *minimum*, *mean minus standard deviation*, *mean*, *mean plus standard deviation* and *maximum* (Figures 3 and 4). Results confirm that the *BM* presents the lowest mean values. Although the standard deviation ranges overlap, it must be noticed that the 95% confidence intervals and paired t-student tests show significant differences.

5 Conclusions and Future Work

The results obtained support the idea that the combination of evolution and learning is an interesting approach, when facing changing environments. Although the simulated environments are commuted pe-

riodically, this information is not used by the proposed learning algorithms, which is useful when changing events are hard to detect. Even when changes can be detected, sometimes these are so rapid that it is not possible to rebuild the learning model.

Referring to the *LM* vs *BM* debate, this work supports the latter when facing changes. One explanation may be that the *BM* is evolving not a final solution, but instead a good set of initial weights.

In the future it is intended to enlarge the experiments to real world domains (e.g. *intensive care units*) and other problem types (e.g. *reinforcement learning*). Furthermore, the learning algorithms could be adapted to perform a simultaneous evolution of neural architectures and weights.

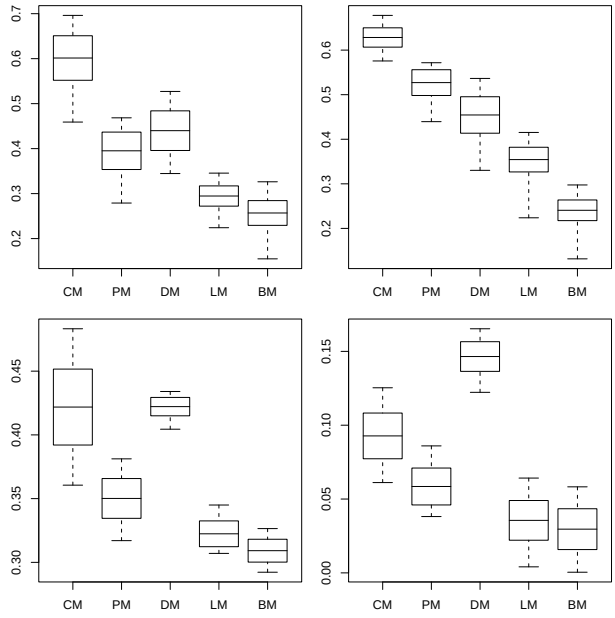

Fig. 3: Box plots for the *soft* experiments ($6BC_S$, TCC_S, CMC_S and CEB_S).

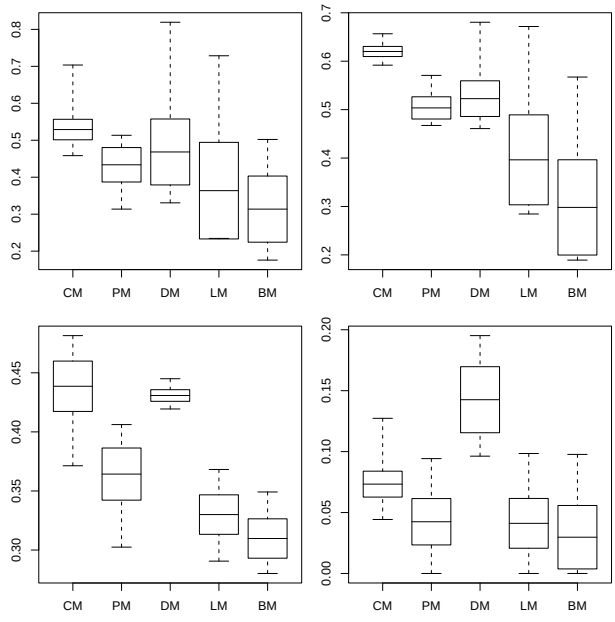

Fig. 4: Box plots for the *hard* experiments ($6BC_H$, TCC_H, CMC_H and CEB_H).

References:

[1] J.M. Baldwin. A New Factor in Evolution. *American Naturalist*, (30):441–451, 1896.

[2] C. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[3] C. Blake and C. Merz. UCI Repository of Machine Learning Databases, 1998.

[4] P. Cortez, M. Rocha, and J. Neves. A Lamarckian Approach for Neural Network Training. *Neural Processing Letters*, 15(2):105–116, April 2002.

[5] M. Goebel and L. Gruenwald. A Survey of Data Mining and Knowledge Discovery Software Tools. *SIGKDD Explorations*, 1(1):20–33, 1999.

[6] T. Lim, W. Loh, and Y. Shih. A Comparison of Prediction Accuracy, Complexity and Training Time of Thirty-three Old and New Classification Algorithms. *Machine Learning*, 40(3):203–228, 2000.

[7] M. Riedmiller. Supervised Learning in Multilayer Perceptrons - from Backpropagation to Adaptive Learning Techniques. *Computer Standards and Interfaces*, 16, 1994.

[8] M. Rocha, P. Cortez, and J. Neves. Evolutionary Neural Network Learning. In F. Pires and S. Abreu, editors, *EPIA 2003 Proceedings, LNAI 2902*, pages 24–28. Springer, 2003.

[9] P. Turney. Myths and Legends of the Baldwin Effect. In *Proc. of ICML96*, pages 135–142. Springer, 1996.

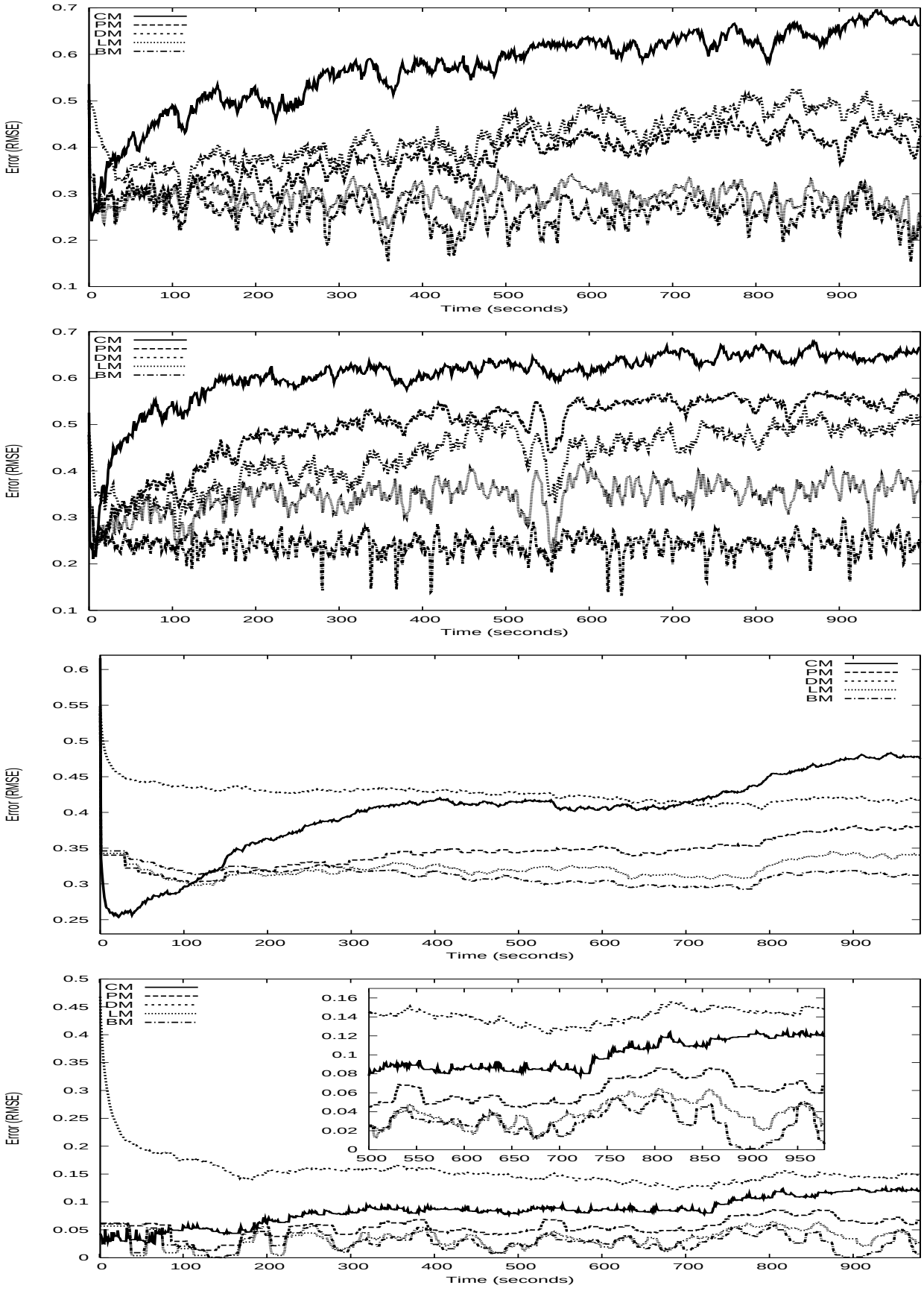

Fig. 5: Results for *soft* experiments ($6BC_S$, TCC_S, CMC_S and CEB_S).

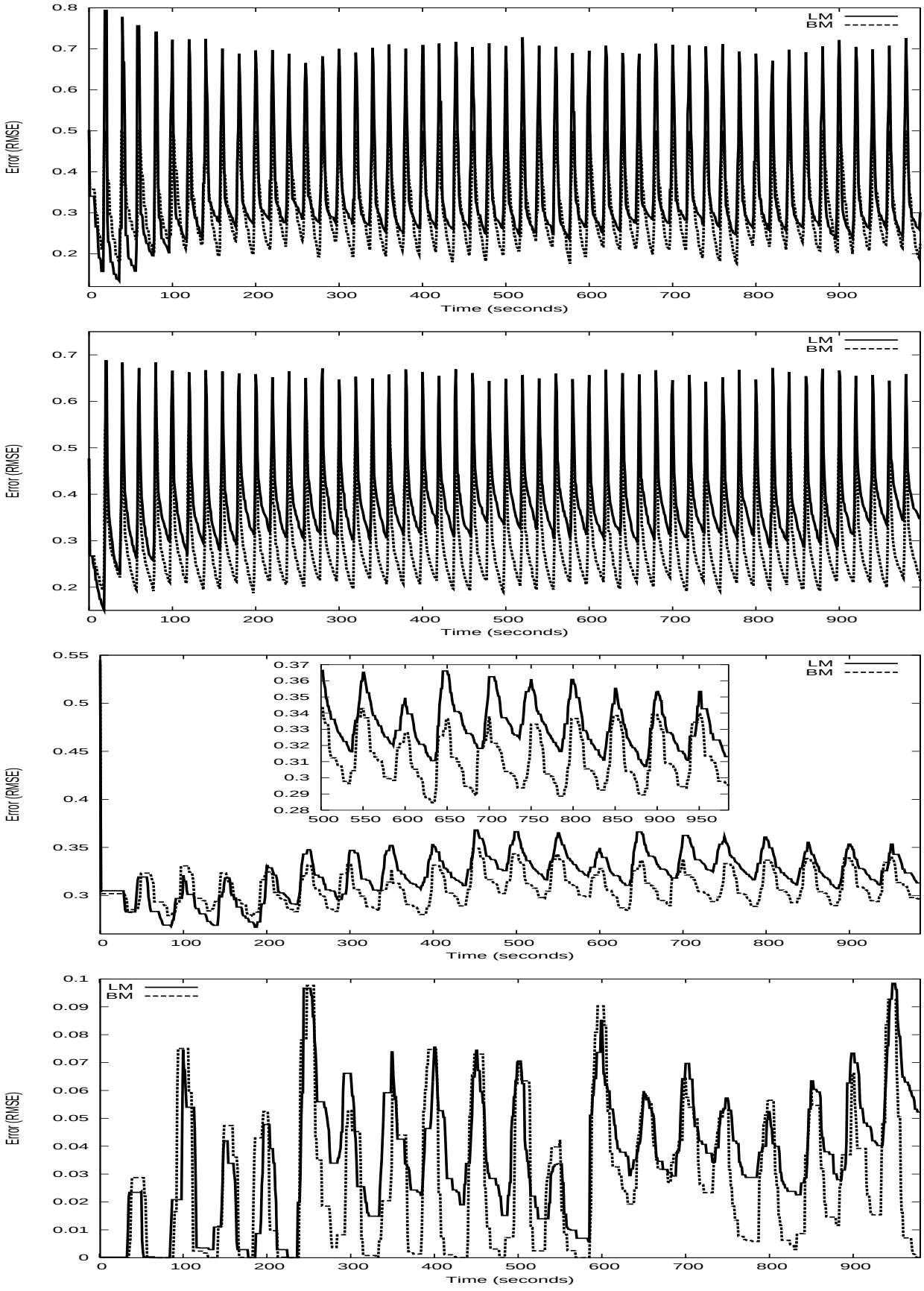

Fig. 6: Results for the *hard* simulations ($6BC_H$, TCC_H, CMC_H and CEB_H).