Neural Network Based Feedback Scheduling of Multitasking Control Systems

Feng Xia and Youxian Sun

National Laboratory of Industrial Control Technology,
Institute of Modern Control Engineering
Zhejiang University, Hangzhou 310027, China
{xia,yxsun}@iipc.zju.edu.cn

Abstract. To cope with resource constraints in multitasking control systems, feedback scheduling is emerging as an important technique for integrating control and scheduling. The ability of neural networks (NNs) as good and robust nonlinear function approximators, reducing the computation time as compared against algorithmic solutions, suggests applying them to the feedback scheduling problem. A novel, simple and intelligent feedback scheduler is presented using a feedforward NN. The algorithmic optimizer is utilized as a teacher to generate data samples for NN training. The role of the NN based feedback scheduler is to provide a good approximation to the optimal solution and online adjust the sampling period of each control task so that the overall system performance is maximized in the face of workload variations. The performance of the NN approach is evaluated through co-simulations of the scheduler, controllers and process dynamics.

1 Introduction

Recently, there is a trend towards integrated control and scheduling [1,2], where both the real-time computing community and the control community are involved. With rapid evolution of embedded computers for real-time control, it is increasingly common that some independent control tasks reside in the same processor, resulting in multitasking control systems. In these cases, the CPU time, which is always limited in an embedded environment, becomes the shared resource that several control tasks compete for. Consequently, the performance of multitasking control systems depends not only on the controller design, but also on the efficient scheduling of the shared computing resources. Traditional control systems are developed without much attention to the resource limitations, which only motivate real-time researchers. As these systems move from the research laboratories to real-world applications, they will inevitably suffer from resource constraints, which give rise to overload conditions in the shared processor. As a result, the performance of the control loops may be greatly degraded, and even destabilized sometimes.

To address these requirements, a few efforts (e.g. [3-5]) have been made in the emerging field of feedback scheduling. When encountering workload variations, the feedback scheduler typically adjusts the sampling periods of concurrent control tasks so that the system remains schedulable and the overall control performance is optimized. However, the system performance is generally nonlinear as a function of the sampling period of each control loop. The optimization routine, which may corresponds to constrained Newton optimization, inside a feedback scheduler can be very time consuming [5]. This is expensive for real-time control, and a computationally cheaper algorithm is desirable. Although approximation versions [3] of the optimal feedback schedulers are suggested in terms of linear proportional rescaling of the nominal sampling periods, they are somewhat empirical, and high accuracy of the resulting solutions cannot always be guaranteed.

In this paper, we present a neural network (NN) approach to the problem of feedback scheduling. Since new tasks can be activated or terminated, and computation time can vary due to the non-deterministic behavior of underlying platforms, the CPU workload may change over time. As one of the advanced technologies of recent times, neural networks [6,7] offer an exciting computational model. The ability of neural networks as good and robust nonlinear function approximators, reducing the computation time as compared with algorithmic solutions, suggests applying them to the feedback scheduling problem. We intend to exploit a simple and effective structure that uses a feedforward NN for feedback schedulers, demonstrating a novel application of NN at the same time. A back propagation (BP) NN trained with the Levenberg-Marquardt (LM) algorithm [6,8] is employed to dynamically adjust the sampling periods of control tasks. The existing algorithmic approach is used as a teacher to label the data samples for the NN training. Once well-trained offline, the NN is used to deliver almost optimal solution to the feedback scheduling problem so that the overall performance of all control loops is maximized whenever the CPU workload changes.

The rest of this paper is structured as follows. The problem of feedback scheduling is formulated in Section 2. We present the NN based feedback scheduler in Section 3. And its performance is evaluated via preliminary simulations in Section 4. In Section 5, we conclude this paper.

2 Problem Formulation

The system we consider is a multitasking control system where a set of controllers shares a CPU with limited computing capacity. As mentioned above, the role of feedback scheduling is to maximize the total control performance under task schedulability constraints. The feedback scheduler should control the CPU utilization at a desired level by dynamically adjusting the controllers' sampling periods. It is assumed that the execution times of all tasks and the workload are available from the real-time kernel at all times. From the control perspective, the scheduler can be viewed as a controller. The CPU utilization corresponds to the controlled variable and the manipulated variables are the sampling periods.

It has been pointed out in [3,5] that the feedback scheduling can be formulated as an optimization problem. Let the number of control tasks within the CPU be n, and let each task execute with a period h_i (sampling period) and have the execution time C_i. Each task is associated with a cost function $J_i(h_i)$, which gives the control cost as a function of sampling period h_i. Then the feedback scheduler should solve the problem

$$\min_h \ J = \sum_{i=1}^{n} J_i(h_i) \quad \text{s. t.} \ \sum_{i=1}^{n} C_i / h_i \le U_{ref} \tag{1}$$

where U_{ref} is the desired utilization level. It can be seen that the minimization routine of feedback scheduling constitutes a nonlinear programming (NLP) problem. To get linear constraints, the costs are recast as functions of sampling frequencies $\mathbf{f} = [f_1,...,f_n]^{\mathrm{T}}$ instead of the sampling periods. Let $V_i(f_i) = J_i(1/h_i)$. According to the Kuhn-Tucker conditions [3,9], if $\mathbf{f}^* = [f_1^*, ..., f_n^*]^{\mathrm{T}}$ is the optimal solution, then

$$\begin{aligned} \nabla\mathbf{V}(\mathbf{f}^*) + \lambda\mathbf{c} &= 0 \\ \lambda[U_{ref} - \mathbf{c}^{\mathrm{T}}\mathbf{f}^*] &= 0 \\ \lambda &\geq 0 \end{aligned} \tag{2}$$

where $\nabla\mathbf{V}$ is the gradient vector, $\mathbf{c} = [C_1, \ldots, C_n]^{\mathrm{T}}$, and λ is the Lagrange multiplier.

Although there are several techniques to solving such a constrained NLP problem [9], e.g. Zoutendijk feasible direction algorithm and Frank-Wolfe method, solving the optimization problem exactly can be very time-consuming [5]. An algorithmic optimizer representatively involves calculation of the derivatives (gradients) or the Jacobian matrix, and a significant number of iterations are often needed. This will cause large computing overhead and hence holdbacks the application of existing NLP algorithms in feedback scheduling. In order for feedback scheduling to be practically useful, it is crucial that the scheduling overhead is relatively small. To address these problems, a simple neural network based approach is proposed in the next section.

3 Neural Network Approach to Feedback Scheduling

The reasons for the use of feedforward NN in feedback scheduling are twofold. First, in contrast to the algorithmic approach, simple feedforward NN based solutions can be delivered in a real-time fashion, reducing the scheduling overhead. Second, the neural networks offer very accurate solutions, e.g., for nonlinear programming problems like feedback scheduling, thanks to their powerful nonlinear approximation capacities. Therefore, we use a feedforward neural network to approximate the algorithmic optimizer and replace it in practical applications. In this case, training data is not a problem since it can be easily created off-line using existing algorithmic techniques. In this section, we describe the network architecture and training methods utilized.

A three-layer feedforward NN (with one hidden layer) is selected as a universal approximator to the NLP optimizer. The execution time C_i of each controller task and the desired utilization level U_{ref} are chosen as the network inputs, while the sampling periods are network outputs, see Fig.1. Accordingly, the numbers of network inputs and outputs are n+1 and n respectively. The principle behind the selection of network inputs/outputs is that all these variables are tightly associated with the CPU utilization level (according to basic real-time scheduling theory), which impacts greatly on the overall control performance of a multitasking control system (according to time-delay systems theory). The relationship between network inputs and outputs is given by

$$\mathbf{Y} = \mathbf{W}_2(\sigma(\mathbf{W}_1\mathbf{X} + \mathbf{B}_1)) + \mathbf{B}_2 \tag{3}$$

where $\mathbf{W}_i$, $\mathbf{B}_i$ are the weight matrices of appropriate dimensions and bias vectors respectively, input vector $\mathbf{X}$=[C_1, …, C_n, U_{ref}], and output vector $\mathbf{Y}$=[h_1, …, h_n].

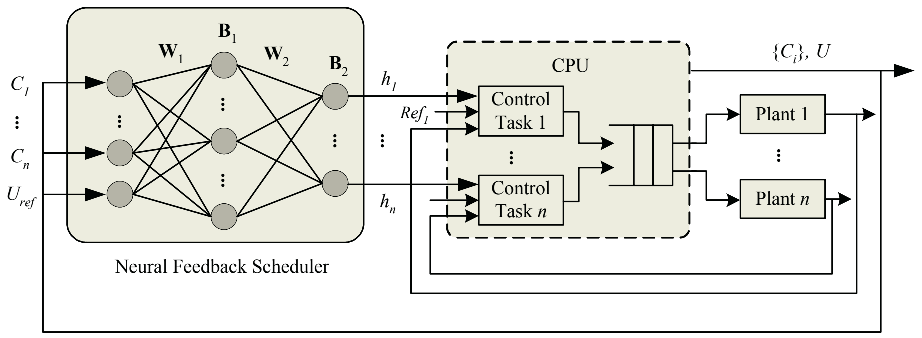

Fig. 1. Neural network based feedback scheduling

As we can see from equation (3), the computational complexity of the NN feedback scheduler mainly depends on the number of hidden neurons and the number of control tasks. In situations with large n, the NN may take a considerable time to yield an output. Fortunately, real-world applications always have limited number of concurrent control tasks that reside in one CPU. Therefore, as long as the number of hidden neurons is chosen properly, small scheduling overhead can be achieved. The activation functions used are the hyperbolic tangent sigmoid transfer function $\sigma(\bullet)$ in the hidden layer and the linear transfer function in the output layer.

To collect data samples for NN training, we solve the feedback scheduling problem with different C_i and U_{ref} as a NLP task. The cost functions are approximated as $V_i(f_i) = \alpha_i + \beta_i / f_i^2$. It has been shown in [3,5] that this is feasible especially for small sampling periods. The algorithmic optimization process may then be conducted using Matlab function *fmincon*. Ideally, the data set for NN training should cover the whole range of the variable execution time of each control task and that of the desired utilization level. After normalized, these data pairs are used as optimal solutions to train and validate the BP neural network. In this work, the neural network is trained offline using the LM algorithm, which appears to be the fastest method for training feedforward networks up to several hundred weights. Since the LM algorithm has an efficient Matlab implementation, we carry out the NN training using the Matlab function *trainlm*.

After it is well trained in supervised mode, the BP neural network can be used to provide a good approximation to the optimal solution for the feedback scheduling problem. Optimal sampling periods, which result in an optimal overall control performance, will be produced with respect to a set of given inputs. In this way, when the allowable CPU utilization or the execution time of a control task changes, the NN feedback scheduler will dynamically adjust the control periods, thus maximizing the overall control performance.

4 Performance Evaluation

We consider simultaneously control two inverted pendulums given by equation (4) with a single CPU, i.e., n = 2. The number of hidden neurons of the NN feedback scheduler is chosen to be 5. Every pendulum is controlled by one LQG controller

[10], which is designed using Matlab and remains the same for different simulations. For comparison, three different cases are considered during simulation studies based on TrueTime toolbox [11]: I) ideal case, where the execution of all tasks are disregarded, II) our solution with NN feedback scheduler, and III) traditional approach without any feedback scheduler. The execution times of both control tasks are varying. In addition to this type of workload variations, we also consider the fluctuation of the desired CPU utilization, which indicates executing or terminating other tasks within the same CPU. What the feedback scheduler concerns is to effectively manage these two types of workload uncertainties and to maximize the overall control performance through optimally adjusting the sampling periods of the two control loops.

$$G_1(s) = \frac{33.6}{s^2 - 33.6}, \quad G_2(s) = \frac{98}{s^2 - 98} \tag{4}$$

In all simulations, the control tasks are assigned fixed priorities. Furthermore, control task 1 is given the lowest priority and will thus suffer most during an overload. In case II, the feedback scheduler is given the highest priority. The execution times of the two control tasks vary in the same range of [8 10] ms, according to the uniform distribution. The nominal sampling periods are 33 and 20 ms respectively. At time t = 2s, U_{ref} changes from 0.8 to 0.5. The data samples for training the neural network is collected as follows: C_1 and C_2 range from 8 to 10 ms with a step of 0.2 ms respectively, while U_{ref} ranges from 0.4 to 1 with a step of 0.025. The corresponding optimal sampling periods are calculated with the help of Matlab. And then the feedforward NN is trained offline. The NN feedback scheduler is implemented as a periodic task with a period of 100 ms and the scheduling overhead is neglected.

Fig. 2 and 3 show the control performance of the two pendulums in different simulations. It is clear that the control performance is degraded when taking into account the execution of the control algorithms, in contrast to the ideal case. From t = 0 to 2s, as we can see from Fig. 2 and 3, two inverted pendulums perform well in all cases. During this time segment, the maximum requested CPU utilization is U_{max} = 10/33 + 10/20 = 0.803 $\approx$ 0.8. Since the available CPU utilization U_{ref} = 0.8, there is sufficient CPU time available for executing both control tasks. This explains the good perform-

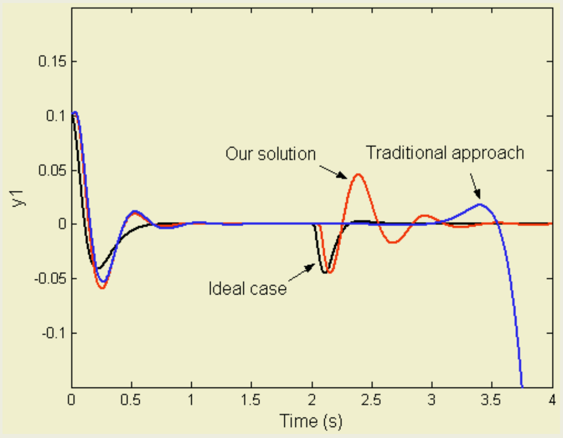

Fig. 2. Transient response of the first inverted pendulum

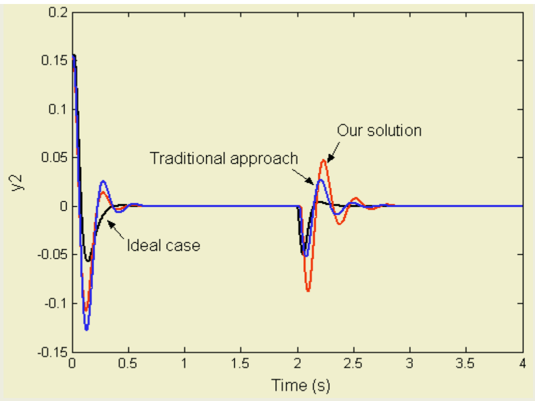

Fig. 3. Transient response of the second inverted pendulum

ance for both pendulums. From t = 2s, the CPU becomes overloaded because the minimum requested CPU utilization $U_{min} = 8/33 + 8/20 = 0.642 > U_{ref} = 0.5$. With traditional approach, the system becomes unstable and the first inverted pendulum falls down, because the control task 1 holds the lower priority. On the contrary, our solution with the NN feedback scheduler provides quite good performance all the time. One can notice that the first pendulum remains stable even when the computing resources become scarce. This is achieved at the expense of slight degradation of the second inverted pendulum's performance. In this way, the overall control performance is maximized. The results demonstrate the practical ability of neural networks to effectively cope with resource limitation and workload uncertainty in multitasking control systems. In the face of overload, graceful degradation of the overall control performance can be achieved online.

5 Conclusions

From the viewpoint of integrated feedback control and real-time scheduling, the real-world resource limitations and workload uncertainties in multitasking control systems are considered. To address these problems, we demonstrate a novel and successful application of neural networks in the newly emerging field of feedback scheduling. The nonlinear approximation capability of feedforward NN is fully exploited. The proposed NN feedback scheduler exhibits several benefits including: 1) as a good approximation to the algorithmic optimizer, it delivers an almost optimal solution; 2) in contrast to the algorithmic approach, it is well-suited for real-time implementation with reduced computational complexity. Therefore, it can be utilized to dynamically maximize the overall control performance of a resource constrained multitasking control system. It is believed that the NN approach will play an increasingly important role in the convergence of control and scheduling in the near future.

References

1. K. E. Arzen, A. Cervin and J. Eker: An introduction to control and scheduling co-design, Proc. of the 39th IEEE CDC, Vol.5 (2000) 4865 - 4870

2. Feng Xia, Zhi Wang, and Youxian Sun: Integrated Computation, Communication and Control: Towards Next Revolution in Information Technology, Lecture Notes in Computer Science, Vol. 3356, Springer-Verlag, Heidelberg (2004) 117-125
3. A. Cervin, J. Eker, B. Bernhardsson, and K.-E. Årzén: Feedback-Feedforward Scheduling of Control Tasks, Real-Time Systems, 23:1 (2002)
4. K. Shin and C. Meissner: Adaptation of control system performance by task reallocation and period modification, IEEE Proc. 11th ECRTS, York, England (1999) 29-36
5. Johan Eker, Per Hagander, KarlErik Årzén: A feedback scheduler for real-time controller tasks. Control Engineering Practice, 8:12 (2000) 1369-1378
6. Hagan, M. T., H. B. Demuth, and M. H. Beale: Neural Network Design, Boston, MA: PWS Publishing (1996)
7. P. K. Campbell, et al.: Experiments with Simple Neural Networks for Real-Time Control. IEEE J. on Selected Areas in Communications, Vol. 15, No. 2 (1997) 165-178
8. Hagan, M. T., Menhaj, M.: Training Feedforward Networks with the Marquardt Algorithm, IEEE Trans. on Neural Networks, Vol. 5, No. 6 (1994) 989-993
9. Mokhtar S. Bazaraa, Hanif D. Sherali, C. M. Shetty: Nonlinear Programming: Theory and Algorithms, 2nd Edition, Wiley (1993)
10. K.J. Åström, B. Wittenmark: Computer Controlled Systems, Prentice Hall (1997)
11. Dan Henriksson, Anton Cervin and K.-E. Årzén,: TrueTime: Simulation of Control Loops Under Shared Computer Resources. In Proc. 15th IFAC World Congress on Automatic Control, Barcelona, Spain (2002)