NEURO-FUZZY CONTROLLER IN REAL-TIME FEEDBACK SCHEDULERS

Tolga AYAV, Prof. Dr. Sinan YILMAZ
Department of Computer Engineering, Faculty of Engineering, Ege University
tayav@likya.iyte.edu.tr, syilmaz@staff.ege.edu.tr

Abstract - *Traditional scheduling algorithms worked on closed and highly predictable environments. However present day systems need to work in more open and unpredictable environments; such as mobile robots, on-line trading, e-commerce, multimedia that cannot be driven well with traditional open-loop algorithms. A new scheduling paradigm, feedback control scheduling, therefore has been presented recently to fulfil the requirements of such systems. This algorithm defines error terms for schedules, monitors the error, and continuously adjusts the schedule to maintain stable performance. When PID (Proportional-Integral-Derivative) controller is used to control the CPU utilization, one of the problems faced is that when utilization setpoint is closer to 100%, in severely overloaded conditions, systems can have a longer settling time than the analysis based on the linear model since utilization feedback saturates at 100%. To overcome this problem, a neuro-fuzzy controller is designed instead of PID. Simulations showed that settling time with the neuro-fuzzy controller is approximately four times shorter than the one with the PID controller.*

I. INTRODUCTION

Real-time scheduling algorithms fall into two categories: static and dynamic. In static scheduling, the scheduling algorithm has complete knowledge of the task-set and its constraints such as deadlines, computation times etc. In dynamic scheduling, however, the scheduling algorithm does not have the complete knowledge of the task-set and its constraints. Dynamic scheduling can be further divided into two categories: scheduling algorithms that work in resource sufficient environments and those work in resource insufficient environments. EDF (Earliest-deadline-first) is an optimal dynamic scheduling algorithm in resource sufficient environments. However in overload situations EDF's performance degrades rapidly. EDF, RM (Rate-Monotonic) and other scheduling algorithms are all open-loop algorithms. Here, open-loop means that; once schedules are created they are not adjusted subsequently based on continuous feedback. While open-loop scheduling algorithms can perform well in static and dynamic systems in which the workloads can be accurately modeled, they can perform poorly in unpredictable systems [1]. Many real-world complex problems such as agile manufacturing, robotics are not predictable. As a cost effective approach to achieve performance guarantees in unpredictable environments, adaptive scheduling algorithms have been developed. While early research on real-time scheduling was concerned with guaranteeing complete avoidance of undesirable effects such as overload and deadline misses, adaptive real-time systems are designed to handle such effects dynamically.

There are many open research questions in adaptive real-time scheduling. In particular, how can a system designer specify the performance requirements of an adaptive real-time system; and how can a designer systematically design a scheduling algorithm to satisfy the system performance specifications. The design methodology for automatic adaptive systems has been developed in feedback control theory [2]. However, feedback control theory has been mostly applied in mechanical and electrical systems. In trying to apply feedback control theory to a computer system domain, the modelling and implementation of adaptive real-time systems face significant challenges. One of these challenges in particular is undertaken in this paper, and relevant solutions are proposed.

II. APPLICATION OF CONTROL THEORY TO SCHEDULING

The mapping of control theory to scheduling provides a systematic and scientific method for designing scheduling algorithms. A typical control system is composed of a controller, a plant to be controlled, actuators, and sensors. It defines a controlled variable which represents the part of the output that is measured and controlled. Setpoint represents the correct value of the controlled variable. The difference between the current value of

the controlled variable and the setpoint is the error. The manipulated variable is the quantity that is varied by the controller so as to affect the value of the controlled variable. The system is composed of a feedback loop as follows:

- The system periodically monitors controlled variable and determines the error.
- The controller computes the required control based on the error.
- The actuators change the value of the manipulated variable to control the system.

In terms of scheduling, the controlled variable can be CPU utilization u(*k*). The manipulated variable must be able to affect the value of the controlled variable. Here, the manipulated variable is the percentage of the optional parts of tasks to be executed. The imprecise computation model tells us that tasks consist of two parts: mandatory and optional. The mandatory part of a task should always be executed before its deadline, however, the optional part may be omitted or postponed if the system is heavily loaded. The utilization of CPU u(*k*) at the k[th] sampling instant is the percentage of CPU busy time in a sampling window (t_k, t_{k-1}). In [2][3], deadline miss ratio was used as the controlled variable. However, here we have chosen CPU utilization since we wish always zero deadline miss ratio and utilization has also a direct linkage with deadline miss ratio.

Our feedback control EDF scheduler consists of a PID controller, an EDF scheduler and a service level controller. Service level controller is the actuator of the system that simply adds the output of the controller to the percentage of the task's optional parts to be executed.

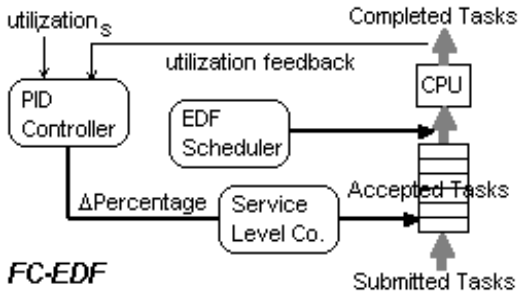

Figure 1. Feedback control EDF Scheduling

The PID controller generates a quantity of change in the percentage of optional parts of tasks to be executed according to the following formula:

$$output = K_p e_n + K_i \sum_{IW} e_i + K_d (e_n - e_{n-1}) \quad (1)$$

One of the problems faced in this figure is that because the utilization of CPU u(k) saturates at 100%, controller cannot detect how severely the system is overloaded when u(k) remains at 100%. Hence, in severely overloaded conditions system can have a longer settling time than the analysis result based on the linear model. The closer the reference is to 100%, longer the settling time will be. This is because the smallness of difference between the setpoint and utilization feedback causes an insignificant change at the output of the PID controller. For example, suppose the total requested utilization is 200% and the setpoint is 99%. In this case, the error measured by the controller would be 0.99-1= -0.01. However, the error should have been 0.99-2= -1.01 according to the linear model. Therefore in such systems, setpoint should be sufficiently far away from 100% to alleviate the impact of saturation on the control performance. Another approach proposed in this paper is to utilize a neuro-fuzzy controller to prevent this non-linearity. Neuro-fuzzy controllers are proved to solve non-linear problems quite successfully.

III. NEURO-FUZZY CONTROLLER

A first order Takagi-Sugeno fuzzy network model with three rules was employed in this work. This network is shown in figure 2. A common rule set for three fuzzy if-then rules is the following:

$Rule_1$:if error is A_{11} and setpoint is A_{12} then y is y_1

$Rule_2$:if error is A_{21} and setpoint is A_{22} then y is y_2

$Rule_3$:if error is A_{31} and setpoint is A_{32} then y is y_3

Where: A_{ij} is a fuzzy set for i[th] rule and j[th] linguistic variable. Note that the error is the difference between utilization and the setpoint. Every membership function in this controller was defined as generalized bell function:

$$\mu_{A_{ij}}(x) = \frac{1}{1 + \left| \frac{x - c_{ij}}{a_{ij}} \right|^{2b_{ij}}} \quad (2)$$

For the application of the rules we need to define a fuzzy inference mechanism. In this case we will take the product operator as T-norm. This means that the firing strength of each rule is

$$u_1 = \mu_{A_{11}} \mu_{A_{12}},\ u_2 = \mu_{A_{21}} \mu_{A_{22}},\ u_3 = \mu_{A_{31}} \mu_{A_{32}}$$

Thus the output of the network is

$$y=\frac{\sum_{i=1}^{n} u_i k_i}{\sum_{i=1}^{n} u_i} = \frac{u_1}{u_1+u_2+u_3} k_1 + \frac{u_2}{u_1+u_2+u_3} k_2 + \frac{u_3}{u_1+u_2+u_3} k_3 = \sum_{i=1}^{3} \bar{u}_i k_i . \quad (3)$$

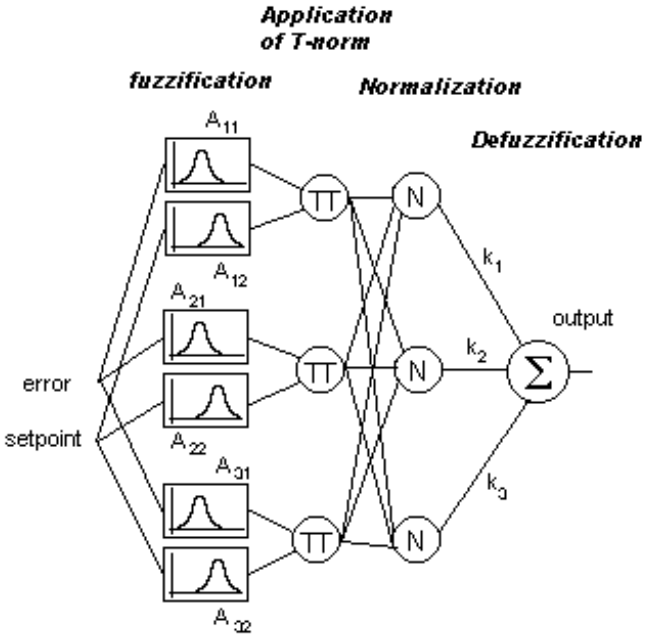

Figure 2. The Neuro-Fuzzy Controller

A. *Learning*

Learning procedure is an approach to the parameter estimation problem and there are several methods described in the literature. The method used in this paper is a supervised learning procedure.

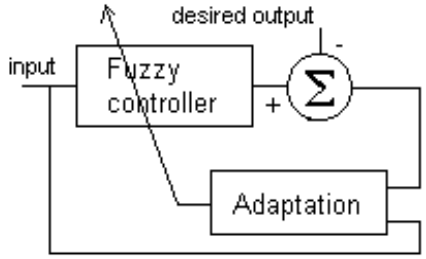

Figure 3. Closed-loop adaptation

Supervised learning procedures are an application of adaptive algorithms in the form of a closed-loop adaptive system as seen in figure 3 [4]. The error signal is the difference between the desired output and the actual output of the system. Using this error, an adaptive algorithm adjusts the parameters of the system, altering its response characteristics by minimizing a measure of the error. The aim is to find an extremum of a criterion (loss) or energy function *V* considered as a function of the parameters of the unknown system. The criterion function is usually defined as:

$V=\frac{1}{N}\sum_{t=1}^{N} e^2(t)$, where N is the number of training patterns. *(4)*

The parameter vector z shown below consists of the parameters of the membership functions (eq. 2).

$$z=[a_{11}, b_{11}, c_{11}, a_{12}, b_{12}, c_{12}, k_1, a_{21}, ..., b_{32}, c_{32}, k_3]$$

Therefore, the iterative learning rule can be defined as:

$$z_k(t+1) = z_k(t) - \eta \frac{\partial V(z)}{\partial z_k} \quad (5)$$

We used the following training data. All except the 17th were obtained directly from the PID controller's output (Kp=0.2, Ki=0.1 and Kd=0) according to equation 1. The 17th pattern is changed as Δoutput gives ten times of the PID's output to reduce the aforementioned settling time.

Pattern No	*Input*		*Output*
	Error: setpoint-utilization	*Setpoint*	*Δ(Reject percent)*
1	-0.2	0.5	-0.08
2	-0.1	0.5	-0.04
3	-0.05	0.5	-0.02
4	-0.01	0.5	-0.004
5	0.01	0.5	0.004
6	0.05	0.5	0.02
7	0.1	0.5	0.04
8	0.2	0.5	0.08
9	-0.2	0.8	-0.08
10	-0.1	0.8	-0.04
11	-0.05	0.8	-0.02
12	-0.01	0.8	-0.004
13	0.01	0.8	0.004
14	0.05	0.8	0.02
15	0.1	0.8	0.04
16	0.2	0.8	0.08
17	**-0.01**	**0.98**	**0.04**
18	0.01	0.98	0.004
19	0.05	0.98	0.02
20	0.1	0.98	0.04
21	0.2	0.98	0.08

Table 1. Training data

Training was stopped after approximately 80,000 iterations when the energy reached to 0.000704.

B. Simulation

First, PID and neuro-fuzzy controllers were compared with setpoints changed from 0.6 to 0.8.

Figure 4 and 5 shows the results. As seen, the designed neuro-fuzzy controller shows a successful similarity to the PID controller.

The other simulation was performed to show the differences of the responses of the two controllers

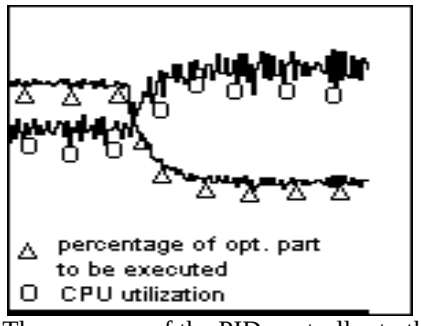

Figure 4. The response of the PID controller to the change in setpoint from 0.6 to 0.8.

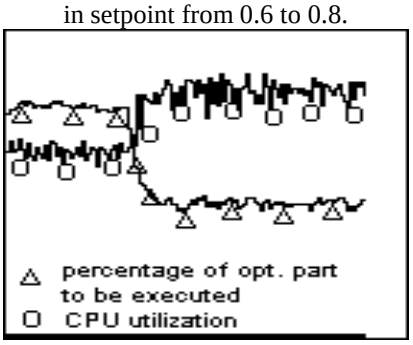

Figure 5. The response of the neuro-fuzzy controller to the change in setpoint from 0.6 to 0.8.

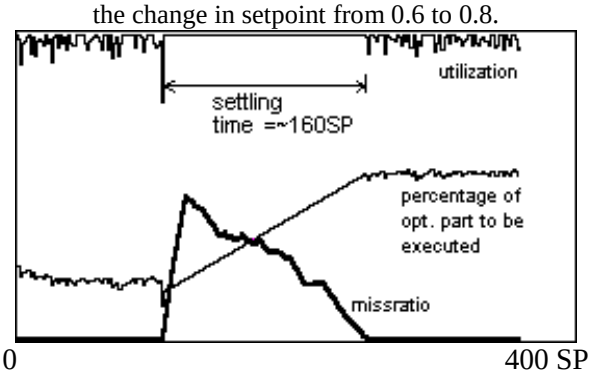

Figure 6. The PID controller's response to the workload change when utilization setpoint is 0.98.

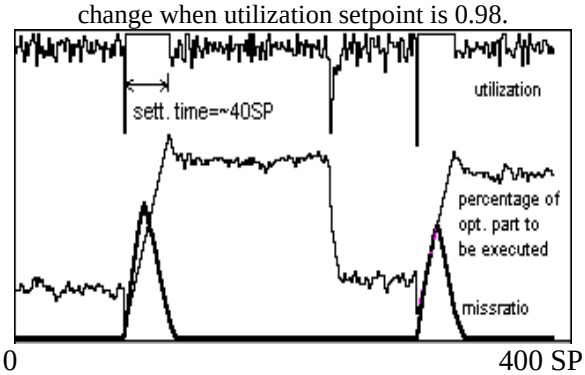

Figure 7. The Neuro-Fuzzy controller's response to the workload change when utilization setpoint is 0.98.

when the setpoint is very close to 100% (i.e 98%). Two different workloads were generated and the system was simulated for workload changes from low to high for both PID and neuro-fuzzy controller.

In the simulation work carried out, the execution times of mandatory parts of tasks were generated from a uniform distribution; in the range of [5, 8] and optional execution times were generated from a uniform distribution [5, 15]. Two workloads differed only with respect to the number of tasks, and not the distributions involved. The first workload has 15 periodic tasks and the second has 20 periodic tasks. There is no aperiodic task and all periodic tasks are independent. The deadline of the tasks are also their periods that are generated from the uniform distribution [200, 250]. SP is the sampling period at which the utilization is monitored and the controller generates an output. It is set to 600 time units. Figure 6 and 7 show the results of the second simulation.

IV. Conclusion

In this paper, we propose a control theory based framework for real-time schedulers. Feedback control real-time scheduler has been an important research issue recently and it still has many open research questions such as: the right choices of controlled, manipulated and other variables, modeling of feedback control system and the necessity of an adaptive form of the PID controller. As stated above, one of the main problems with the PID controller is the degradation of the controller's performance in the case of utilization feedback being too close to 100%. To overcome this problem, a neuro-fuzzy controller was designed instead of PID and simulations showed that the neuro-fuzzy controller is as successful as PID and has a shorter settling time when the setpoint is close to 100%.

References

[1] Chenyang Lu, John A. Stankovic, Gang Tao, Sang H. Son, "Design and Evaluation of a Feedback Control EDF Scheduling Algorithm".

[2] Chenyang Lu, John A. Stankovic, Gang Tao, Sang H. Son, "Feedback Control Real-Time Scheduling: Framework, Modeling, and Algorithms", Proc. Of IEEE Real-Time System Symposium.

[3] D.A.Lawrence, J.Guan, S.Mehta, L.R. Welch, "Adaptive Scheduling via Feedback Control for Dynamic Real-Time Systems", Proc. Of 2001 IEEE.

[4] Jelena Godjevac, Nigel Steele, "Neuro-fuzzy control of a mobile robot", Neurocomputing'1999, Elsevier.