5-2019

Analysis Literatures of Machine Learning and Neural Networks for Real Time Scheduling

Phong Nguyenho
University of Central Florida, bigbox123@knights.ucf.edu

Mark Nguyen
University of Central Florida

Phong Nguyenho, Mark Nguyen
EEL 4932

Analysis Literatures of
Machine Learning and Neural Networks for Real Time Scheduling

Introduction

Real time scheduling problems are present in every aspect of software development. An optimized real time scheduling scheme would determine the performance of an operating system. There are many different approaches that real time scheduling researchers developed to tackle scheduling problems in many computer systems that have great important roles in keeping our modern society running smoothly. Neural-network real time scheduling is one of those approaches that can solve many computer scheduling problems. As computing technology advanced, more and more real time scheduling problems arise that need new solutions to keep up with the demand of faster computer systems. In this literature review, we analyze four research papers that promote some great solutions for some particular scheduling problems. The first one is "*A Neurodynamic Approach for Real Time Scheduling via Maximizing Piecewise Linear utility"* by Zhishan Gou and Sanjoy K. Baruah (2016). The second paper is "*Scheduling Multiprocessor Job with Resource and Timing Constraints Using Neural Networks"* by Y. Huang and R. Chen (1999). The third paper is "*Solving Real Time Scheduling Problems Using Hopfield-Types Neural Networks"* by M. Silva, C. Cardeira, and Z. Mammeri (1997). Finally, the last one is "*Neural Network for Multiprocessor Real Time Scheduling"* by C. Cardiera and Z. Mammeri (1994).

A Neurodynamic approach for Real Time Scheduling via Maximizing Piecewise Linear Utility

This research paper introduces a new algorithm based on the combination of neurodynamic optimization with real time scheduling. The study focuses on real time utility maximization scheduling problem on uniprocessor systems. The organization of this paper is very easy to follow. Each section is dedicated to describe a process of the study. Section I is the introduction which thoroughly introduces the set of problems that need to be solved, the constraints that require to be fulfilled in solving those problem, and the proposed solution that is inspired by prior works of other researchers. Section II is the description of the problem set that is accompanied by 2 examples to demonstrate the problem. Section III is the description of the new approach that would solve the problem. Section IV is the description of experimental study for the proposed approach and the comparison between this approach and some traditional approach. Section V is the description of the analyzation of the proposed approach and possible improvements. Section VI is the last section that concludes the result of the study and suggest more direction for further study.

The introduction of this paper gives very detailed information about the problem and the groundworks that inspire and lead to this study. This section has some sub sections that further explain the initial problem of real time scheduling, the constraints, the proposed approach with related work, and the insight of the paper organization. Real time scheduling is the timing requirement that is often modeled by deadlines. An optimized real time scheduler has to ensure that the scheduled task set does not have any task that misses its deadline. Utility maximization is an additional constraint with timing constraints for a real time scheduler. Utilization of the resources that is allocated to the scheduled task set has to be maximized to ensure optimal performance. Neurodynamic optimization is the proposed approach that was somewhat inspired by the advancement of recurrent neural network (RNN) based approaches through studies in the past two decades. The related work of previous studies on neural network only solve specific problem based on certain assumption, so it is hard to extend them for scheduling purposes.

However, in the past two decades, the efforts of the computational intelligence community has pushed some boundaries and achieved some successes.

The second section of the paper is the model and problem description. Utility function in this study is restricted to be piecewise linear, and it is used to replace the deadline in the model. Problem description consists of hard deadline and overload condition, mixed criticality, and NP hardness. In hard deadline and overload condition, an optimized scheduler is not only keeping all the deadlines met but also reduce the cost of time when the system in overload condition. The assignment of different important levels of jobs in mixed critical scheduling helps all the high important jobs met their deadlines before the rest. NP hardness is proved to be consistent in oveload case.

The neurodynamic approach is approximately solve the NP hard problem of real time scheduling by modeling the problem in RNN model and deriving the new algorithm. Instead of using gradient information to guide the stationary data points, this approach uses RNN model to deal with the linear constraints of the problem. After the RNN model minimized the problem's linear piecewise property while the utilization is maximized, the scheduling algorithm is derived as the following steps: first step is compute the intervals of release times and deadlines, second step is compute the values of the piecewise linear intervals, the third step is apply the RNN model:

$$\epsilon \frac{d\mathbf{x}}{dt} = C^T g_{[l,h]}(C\mathbf{x} - \mathbf{f}) - \sigma A^T g_{[0,1]}(A\mathbf{x} - \mathbf{b});$$

with large enough sigma:

$$\sigma \geq \frac{\max_{\xi | \xi \in \Pi_{i=1}^{p} [l_i, h_i]} \|C^T \xi\|_2}{\min_{\gamma | \forall i, \gamma_i \in [0,1]; \exists j, \gamma_j = 1} \|A^T \gamma\|_2}.$$

Section IV is the description of the experiment. This section consists of 3 sub sections: an example of overload case, the comparison of the between the new approach and some typical approaches, and the analyzation of converging time. The overload example is used to further demonstrate the superior of the proposed approach and show the large scale of the experiment. The benchmark for comparison is comparing the gains of utilization over the total weight $(\sum_i w_i).$ The approximation ratio is compared against RNN based method, EDF, Robust EDF, and Fixed Priority. The converging time of the proposed approach remains about the same level when the number of sample size varied.

Section V and VI of the paper consist of further analyzing the proposed approach optimality, possible improvement and the conclusion. Further analyzing results that the proposed approach is optimal to the NP hard problem. The boundary of worst case is derived with the approximation ratio of 0.5 when the overloaded case is online scheduling and release times remain unknown until they become active. Possible improvement to the approach includes minimization of response time and minimization of soft deadline and tardiness. The scope of this paper only covers hard deadline and utility maximization; therefore, the other constraints may not be optimized, and it opens for extensions. The conclusion wraps up the paper with the summaries of the previous sections.

Scheduling Multiprocessor Job with Resource and Timing Constraints Using Neural Networks

The Mean Field Annealing algorithm (MFA) was derived from the Simulated Annealing (SA) method. SA was used to eliminate the deterministic feature that the Hopfield Neural Network had. MFA uses the mean field approximation technique which contains the attributes of both the Hopfield Neural Network method and the Simulated Annealing method (Huang & Chen, 1999). This algorithm provides a means to solving problems that involve combinatorial optimization such as the traveling salesman problem or the minimum spanning tree problem. Huang and Chen (1999) found that the normalized MFA can be broken down into three steps, the initial average state values are randomly set and the highest temperature is used to start, go through the sequential iterations using an energy function and normalization equation that they derived, and decreasing the annealing temperature while still going through the iterations until the convergence has been reached. There are constraints that must be considered when using this method, first, the execution time for each job is predetermined, second, the jobs can be broken into segments with preemptive execution, and third, there is no job migration (Huang & Chen, 1999). These constraints can then be categorized into two types, the output state constraint, deadline and resource constraint (Huang & Chen, 1999).

From Huang and Chen's (1999) simulation results, they found that the Hopfield Neural Network oscillates to convergence compared to the normalized MFA, which has a smooth convergence. For their simulations, they used a variety of initial neuron states and two sets of timing and resource constraints, they scheduled this twice, first with four jobs (processes) and two machines (processors), and the second with five jobs and two machines (Huang & Chen, 1999). The initial temperature used for the MFA simulation is ten and it ran for 200 sequential iterations while decreasing the temperature with every iteration (Huang & Chen, 1999). The results are then displayed on a Gantt chart to show the job schedules with an energy curve. The energy function used for the MFA curve is

$$E = \frac{C_1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{k=1}^{T}\sum_{i1=1}^{N} V_{ijk}V_{i1jk} + \frac{C_2}{2}\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{k=1}^{T}\sum_{j1=1}^{M}\sum_{k1=1}^{T} V_{ijk}V_{ij1k1} + \frac{C_3}{2}\sum_{i=1}^{N}(\sum_{j=1}^{M}\sum_{k=1}^{T} V_{ijk} - P_i)^2 + \frac{C_5}{2}\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{k=1}^{T} V_{ijk}G_{ijk}^2 H(G_{ijk}) +$$

$$\frac{C_6}{2}\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{k=1}^{T}\sum_{i1=1}^{N}\sum_{j1=1}^{M}\sum_{s=1}^{F} V_{ijk}R_{is}V_{i1j1k}R_{i1s}$$

, where i1 ≠ i and j1 ≠ j (Huang & Chen, 1999). The constant weight factors that Huang and Chen (1999) have determined for the MFA energy function are C1 = 6.0, C2 = 6.0, C3 = 5.0, C5 = 3.0, and C6 = 6.0.

The time complexity of this algorithm is O($N^2 * M^2 * T^2$) where N is the total number of jobs to be scheduled, M is the total number of machines to be operated, and T is the deadline of the job (Huang & Chen, 1999). The time complexity was derived from the fact that the computational time for each neuron (N * M * T) is multiplied with the consuming time for each iteration which is equal to the total neuron (N * M * T) (Huang & Chen, 1999). Although the time complexity is not linear, it can still be improved upon so that the time complexity can be reduced. An idea to think about is how to reduce the number of jobs to be scheduled in each neuron for each iteration.

Solving Real Time Scheduling Problems with Hopfield-type Neural networks

Real time scheduling problem has become more complex with faster computer systems. Many researchers have been continuingly looking for new approach that can solve this ongoing scheduling problem. This paper proposes the approach of using Hopfield-type neural networks to solve complex combinatorial problems of real time scheduling. The paper consists of 5 sections. The first section is the introduction. The second section is the summaries of the main characteristics of Hopfield-type neural networks. Procedure of how to utilize the proposed approach to solve real time scheduling problem takes place in section 3. Section 4 consists of digital simulation and complexity analysis. Finally, section 5 is the paper's conclusion.

The first part of the introduction gives the information about the researchers' realization of the fast problem solving potential of Artificial Neural Networks (ANNs), but these neural networks can also produce sub-optimality and un-feasible solutions. However, they developed a mapping method that can use Hopfield-type neural networks to solve the problems of real time scheduling. The second part of the introduction gives a brief insight of the new method's requirements and its ability to solve the problem. The last part of section 1 gives the structure of the paper.

Section 2 consists of the summaries of Hopfield-type neural networks main characteristics. Hopfield-type neural networks is a type of neural networks that their output vector V is guided by the neural networks' dynamics through some variations of the bounded function E:

$$E(v) = -\frac{1}{2} v^T T \; v - v^T i^b$$

where $v_i \in [0,1]$.

Different dynamics minimize the function E along the dynamics' evolution. The continuous Hopfield network dynamics is modeled as the following linear differential equations:

$$\dot{u} = -\eta u + Tv + i^b$$
$$v_i = g_i(u_i)$$

where: v - output vector; u - state vector; T - matrix of weights; i^b - input bias vector; η - diagonal matrix with the inverse of the neurons' time constants;

The third section gives the procedure of how to map the scheduling problem into neural networks. This section has 4 sub sections that shows the steps of the process. The first step is transform the a task schedule into binary form, so it will be easier to put into neural network architecture. The second step is transposed the binary form of the task schedule in a form of quadratic function: $A^{in} v \le b^{in}$

This step also includes some constraints as hardware constraint for processor exclusion to handle inequality constraints, timing constraints. The third step is the mapping process. The goal of this step is to construct function E so that the network solution not only minimize:

$$E^{op}(v) = -\frac{1}{2} v^{+T} T^{op+} v^{+} - v^{+T} i^{op+}$$

but also satisfies the problem's constraints. The fourth step is forcing convergence to a binary solution. The procedure that is used to force convergence to a vertex is called annealing technique. The convergence is guaranteed if the following function is concave:

$$E^{op} = -\frac{1}{2} \sum_{i=1}^{n} v_i(v_i - 1) = -\frac{1}{2} v^T I v - \left(-\frac{1}{2} o^T v\right)$$

where o is a vector of ones.

Section 4 of the paper are the simulation results and complexity analysis. The results shows that on about five time constraints, a few ten of microseconds, the network is stabilized. The result is very promising and encouraging to open up study in more scheduling algorithms using Hopfield-type neural network. The researchers state that the complexity analysis should be done referring to time necessary to calculate neural network time programing. As of the expected network stabilization time is a few characteristic times of the neuron, if the analog hardware implemented.

Section 5 is the conclusion that summarizes the whole process of the study. The algorithm is optimal with respect to the scheduling problem constraints of multiprocessor preemptable periodic tasks.

The conclude the paper with the suggestion that the prospective is very attractive if people continue the development of this technique to online scheduling.

Neural Networks for Multiprocessor Real-Time Scheduling

The Neural Scheduler Algorithm (NSA) is an evolved form of the Hopfield Neural Network and is built around solving constraints satisfaction problems rather than solving problems involving combinatorial optimization. Cardeira and Mammeri’s goal was to derive an energy function from the Lyapunov function, analyze the complexity of algorithm based on that function, and test its quality. They used the following steps from the Hopfield Neural Network, find a neural network topology, find an energy function for the network, and calculate the neuron inputs and weight constants from the energy function (Cardeira & Mammeri, 1994).

In figuring out the constraints and rules for this algorithm, Cardeira and Mammeri (1994) disregarded the number of processors used because that did not increase the number of neurons. They were instead interested in configurations of tasks such as the execution time (c), the deadline (d), and the ready time (r). To meet the timing constraints they considered the tasks to start after the ready time and finish before the deadline. Then they applied the k-out-of-N rule for each timing constraint and sum the new weights and neuron inputs to existing ones (Cardeira & Mammeri, 1994). Finally they applied the P-out-of-T rule for the neurons of the same column because they considered that the number of tasks cannot be greater than the number of machines and if the machine is fully utilized then the number of active neurons for each time unit is equal to the number of machines (Cardeira & Mammeri, 1994).

In Cardeira and Mammeri’s results (1994), the algorithm that they presented has a very fast convergence rate, but it could fall into local minima of the energy function. An observation that Cardeira and Mammeri (1994) made was that the algorithm converges after a few iterations which means execution time is negligible. The energy function that they derived from the Lyapunov function is $E = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(-2)x_i x_j - \sum_{i=1}^{n}(2k-1)x_i$, where j ≠ i. In their tests, they use three tasks with time constraints, the first task has a period of 6 and execution time of 2, the second task has an execution time of 3 and cannot begin to execute before the 2nd time unit and cannot finish later than the 6th time unit, and the third task has an execution time of 1 and release time at 0, but it must finish before the 4th time unit (Cardeira & Mammeri, 1994). They then applied the k-out-of-N rule to these tasks after multiplying them by six time units. Then after the calculations, they are left with several possible solutions. They chose a solution based on the initial state of the neuron, the order in which they are executed, and the amount of noise on the circuit when they implemented the algorithm (Cardeira & Mammeri, 1994). The order in which the neurons are executed are not deterministic because they are computed in parallel and asynchronously and they are implemented by analog amplifiers (Cardeira & Mammeri, 1994).

Cardeira and Mammeri (1994) found that the number of iterations are almost independent of the number of tasks which means that with an increase of neurons, the execution time is unaffected. The time complexity of this algorithm is calculated by taking into consideration the computational time for each neuron O(T * L) and multiplying it with the consuming time for each iteration O(T * L) which then becomes O($T^2 * L^2$) (Cardeira & Mammeri, 1994). The variable T is the number of tasks and the variable L is the number of time units (scheduling length). This time complexity can be further reduced to an (T + L) instead of (T * L) if we take into consideration that each neuron has connections to the neurons in the same row and the same column (Cardeira & Mammeri, 1994). The time complexity only applies to implementations by an analog computer as the advantage of finding fast solutions may be lost when switching to a digital implementation, but considering the fact that the paper was written in 1994 and technology has advanced since then, this advantage will probably not be lost. The time complexity may be improved after switching to a digital implementation.

Conclusion

Real time scheduling algorithms are important as more problems arise because technology is advancing at an incredible rate and the demand for optimal schedulers keep increasing. Neural-network real time scheduling is an approach that can solve the problems that arise from the ongoing demand of faster, more accurate schedulers. The goal of this literature review is to study different real time scheduling algorithms that are based on neural network framework and learn how they are derived from or improved from other approaches. As the authors suggest that these scheduling algorithms can still be improved upon, and there are much more potential in developing better approached to solve the real time scheduling problem based on the framework of machine learning and neural networks.

References

Cardeira, C. & Mammeri, Z. (1994), Neural networks for multiprocessor real time scheduling, IEEE Sixth Euromicro Workshop on Real-Time Systems (ECRTS), 1994.

Guo, Z. and Baruah, S. (2016), A Neurodynamic Approach for Real-Time Scheduling via Maximizing Piecewise Linear Utility, IEEE Transactions on Neural Networks and Learning Systems (TNNLS), vol. 27, no. 2, pp. 238-248, 2016.Y.

Huang, Y. & Chen, R. (1999), Scheduling multiprocessor job with resource and timing constraints using neural networks, IEEE Trans. Systems, Man, and Cybernetics, Part B: Cybernetics, vol. 29, no. 4, pp. 490-502, 1999M.

Silva, M., Cardeira, C., & Mammeri, Z. (1997), Solving real-time scheduling problems with hopfield-type neural networks, IEEE Proceedings of the 23rd EUROMICRO Conference on New Frontiers of Information Technology, 1997C.