2019

Machine Learning and Neural Networks for Real-Time Scheduling

Daniel Hureira
University of Central Florida, dhureira@knights.ucf.edu

Christian Vartanian
University of Central Florida, vartanian@knights.ucf.edu

Machine Learning and Neural Networks for Real-Time Scheduling

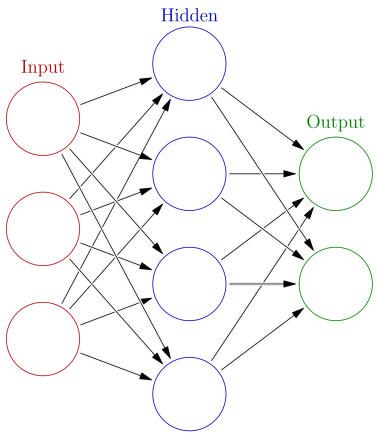
Daniel Hureira & Christian Vartanian

I. Abstract

This paper aims to serve as an efficient survey of the processes, problems, and methodologies surrounding the use of Neural Networks, specifically Hopfield-Type, in order to solve Hard-Real-Time Scheduling problems. Our primary goal is to demystify the field of Neural Networks research and properly describe the methods in which Real-Time scheduling problems may be approached when using neural networks. Furthermore, to give an introduction of sorts on this niche topic in a niche field. This survey is derived from four main papers, namely: *"A Neurodynamic Approach for Real-Time Scheduling via Maximizing Piecewise Linear Utility"* and "*Scheduling Multiprocessor Job with Resource and Timing Constraints Using Neural Networks"*. *"Solving Real Time Scheduling Problems with Hopfield-type Neural Networks"* and *"Neural Networks for Multiprocessor Real-Time Scheduling"*.

II. Hopfield-Type Networks

Neural Networking has become an ever growing necessity in today's ever growing world of complex computing. Mimic the functions of the brain, a Neural Network is a framework for utilizing nodes (neurons) to better increase computational efficiency. Neural Networks take many forms, such as the unidirectional "Forward Flow" Network (Fig 1.) or the feedback inducing "Recurrent Network". Forward Flow networks are generally simple and excel at pattern identification. Their downside is a lack of nodal memory.

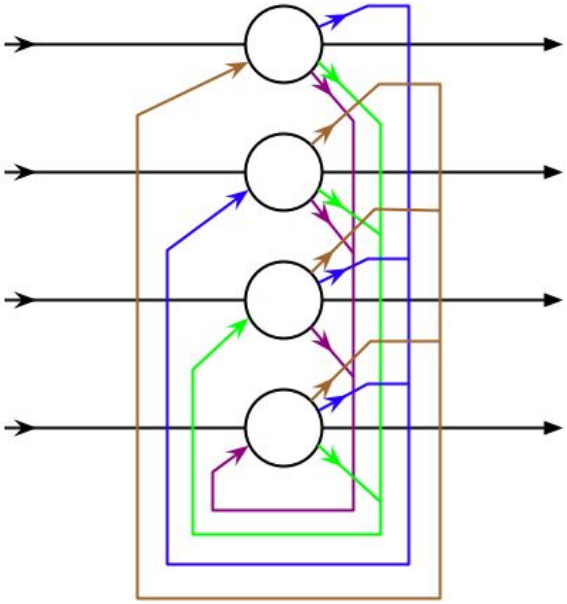


(Fig 1.) A Forward Flow Network

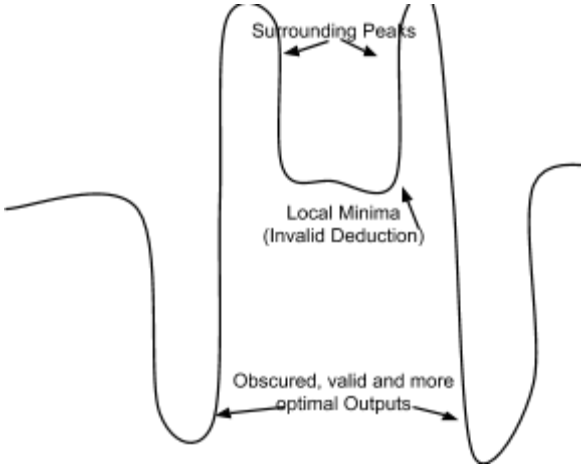
Memory can be gained at each neuron by feeding its output back into itself, thus allowing for a network to learn about the data it is processing unsupervised. When dealing with scheduling problems (or classically, the Traveling Salesman Problem), some kind of memory is required to be able to properly process the input data. As such it becomes apparent that the optimal topology for solving such a problem would be a recurrent network.

An suitable recurrent candidate for solving Hard-Real-Time Scheduling problems is the Hopfield Network. Hopfield-Type Networks are a form of recurrent Neural Network that mimics the operations of human memory. They use binary neurons to simply store state space. The core advantage of Hopfield Networks is their *loose* guarantee of correct convergence. When trained, they are given a minimum energy for each neuron and will, regardless of input, converge back to the local minimum. This convergence allows the networks to recover from distortions, and generally increases their adaptability, giving them a huge benefit in solving real time dynamic problems.

While at first glance the Hopfield Network's tendency to converge seems to be a perfect tool for solving HRT problems, simple analysis shows that there are some issues that arise when testing these networks. Essentially, the Hopfield Network will often times incorrectly conclude that its final destination is the energy minimum (and thus a valid output) when in reality there are many other yet unsearched combinations that yield a closer and potentially valid output. In their 1994 paper, Cardeira and Mammeri provide a simple metaphor for this problem in which a person is searching for the lowest point in a mountain range and incorrectly deduce that the valley that they have found themselves in to be the lowest point. The person makes this conclusion while their sight has been obscured by the surrounding high mountains, hiding the fact from them that there exists far deeper valleys and that they are actually not in a deep valley at all (Cardeira, Mammeri). For Hopfield Networks to be reliably implemented in the search for HRT Schedules, a way of correcting this local minima issue must be developed.



(Fig 2.) A Recursive Network of the Hopfield-Type



(Fig 3.) A Graphical Representation of the Local Minima Problem

III. Getting Over the Hump

The core goal of solving any Hard-Real-Time (HRT) scheduling problem is ensuring that tasks are executed and completed by the deadline. It stands to reason that a neural network would be able to excel at multiprocessor scheduling tasks due to their optimal implementation as search functions. Since the process of scheduling is, at its core, a parallel search tree, applying Neural-Network methodologies to the scheduling process should be an optimal choice. There is one important caveat, however, *speed*.

While optimal over humans at parsing through big data, many Neural-Networking implementations come with high costs in terms of speed. Many networks need to be rigorously trained in order to be able to effectively detect patterns and a methodology of training must be implemented along side a suitable network topology if one is going to optimally use a Neural-Network to schedule task sets. Silva, Cardiera, and Mammeri (SCM) proposed a methodology for training analog Hopfield-Type Networks that sufficiently lowered the processing and training time to a point at which the search process became swift enough to implement for HRT problems.

The SCM Method for training these Hopfield Networks was developed to optimize the initialization process as much as possible. Thankfully, HRT schedules adhere to a *hard* deadline constraint and thus any wrong outputs produced by the network are subject to a binary review of being fully correct or fully incorrect. The method reduces the given problem of task scheduling to that of solving a quadratic optimization problem. Timing Constraints are able to be represented as linear equalities and, when used in conjunction with linear inequalities representative of hardware constraints, the method is able to fully model a preemptible periodic task schedule in a form that is easily solved via a Hopfield-Type Network.

SCM also provides for a verification stage in the scheduling process. A common issue of these network types is a slight tendency to converge at an invalid output (the local minima problem described in the previous section), especially when operating at max load. Luckily, there exists mapping methodologies that are able to stop the occurrence of these errors, forcing all outputs to validly conform to the given constraints. Without this verification enabled mapping, using these Networks would be worthless as the large output schedules would be able to mystify any possible errors due to their size. As such, increasing guarantees of system accuracy are crucial to being able to trust this method for task scheduling.

Upon implementing the SCM on a set of simulated Op-Amp Neurons, the advantages become readily apparent. Multi-Processor Tasks were able to scheduled as quickly as a few tens of microseconds (Silva). At this speed one is easily able to implement Hopfield Networks for use in dynamic real time scheduling.

IV. Further Implementation

A. Uniprocessor Systems

Neural Networks have been used to solve a multitude of problems in the past and to optimize engineering applications. Problems such as the Traveling Salesman Problem (TSP) uses a Hopfield neural network in order to efficiently solve it. Therefore, the same should be able to be accomplished to optimize real-time scheduling problems. Furthermore, neural networks can be used to solve problems regardless of the magnitude of the problem. Therefore, real-time scheduling problems can be solved regardless of the size of the problem.

On this subject, there has been a plethora of research done on neural networks and real-time scheduling problems. However, this research only looks at solving certain real-time scheduling problems under specific assumptions. Therefore, this research cannot be extended to general real-time scheduling. There has been hopes to extend the usefulness of neural networks and real-time scheduling to solve general scheduling problems by using piecewise linear functions.

In order to solve general hard real-time scheduling problems, it is first necessary to convert the modify the constraints and the objective function such that dynamic system converged to a global optima (Guo, Baruah). The following neural network model has been derived:

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

(Fig 4.) A RNN Model used to solve real-time scheduling problems while optimizing utility
(Guo, Baruah)

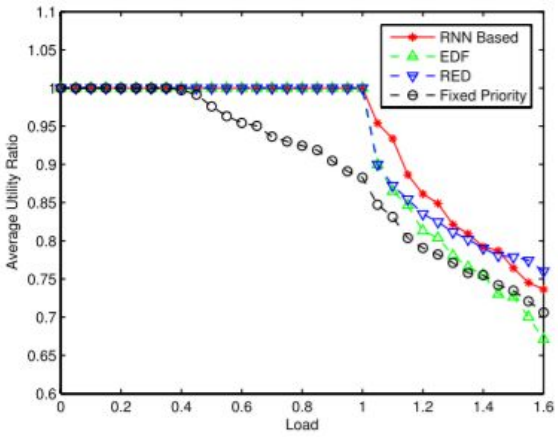
Where the epsilon and alpha represent the positive scaling parameters. Using this neural network model, a large enough alpha is found to satisfy this inequality:

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

(Fig 5.) An optimal solution to hard real-time scheduling problems.
(Guo, Baruah)

The scheduling algorithm performed first calculates the number of intervals, then calculates the constants, applies the neural network model with an alpha large enough to satisfy the optimal solution, record the state variable, and lastly transform the state variable to the appropriate form.

Comparing the neural network model with other classical scheduling algorithms, it can be easily shown that the neural network scheduling algorithm outperforms the other scheduling algorithms. This is shown by comparing the utility of each scheduling method. Below is the graph showing this:



(Fig 6.) Utility comparisons
(Guo, Baruah)

Neural network models have the potential to greatly increase the computing power of modern day devices. Furthermore, it may be able to be ported onto hardware. This would get rid of the current bottleneck in modern day systems of memory and computing power.

B. Multiprocessor Systems

Neural networks can also be applied to multiprocessor systems. This way we can look at jobs being executed on multiple machines. The research done regarding this topic involve using more constraints and deriving an energy function and then

applying neural networks to optimize energy consumption.

Furthermore, many constraints are specified in this problem description. Firstly, they state that the execution times are predetermined. Meaning, that the execution times cannot change mid-way through the execution. Second, preemption does occur in these jobs. And lastly, job migration does not occur. This means, that one job cannot occur on one machine and then move to another machine. With these constraints, research is done in order to optimize the cost and minimize the energy consumption of the machines. Shown below is the energy function derived:

$$
\begin{aligned}
E = & \frac{C_1}{2} \sum_{i=1}^{N} \sum_{j=1}^{M} \sum_{k=1}^{T} \sum_{\substack{i1=1 \\ i1 \neq i}}^{N} V_{ijk} V_{i1jk} \\
& + \frac{C_2}{2} \sum_{i=1}^{N} \sum_{j=1}^{M} \sum_{k=1}^{T} \sum_{\substack{j1=1 \\ j1 \neq j}}^{M} \sum_{k1=1}^{T} V_{ijk} V_{ij1k1} \\
& + \frac{C_3}{2} \sum_{i=1}^{N} \left(\sum_{j=1}^{M} \sum_{k=1}^{T} V_{ijk} - P_i \right)^2 \\
& + \frac{C_4}{2} \sum_{j=1}^{M} \sum_{k=1}^{T} \left(\sum_{i=1}^{N} V_{ijk} - 1 \right)^2 \\
& + \frac{C_5}{2} \sum_{i=1}^{N} \sum_{j=1}^{M} \sum_{k=1}^{T} V_{ijk} G_{ijk}^2 H(G_{ijk}) \\
& + \frac{C_6}{2} \sum_{i=1}^{N} \sum_{j=1}^{M} \sum_{k=1}^{T} \sum_{\substack{i1=1 \\ i1 \neq i}}^{N} \sum_{\substack{j1=1 \\ j1 \neq j}}^{M} \sum_{s=1}^{F} V_{ijk} R_{is} V_{i1j1k} R_{i1s}
\end{aligned}
$$

(Fig 7.) Energy function
(Huang, Chen)

Where C_n refers to constants.

This equation is then applied to two neural network algorithms, the Hopfield neural network and the MFA network. Afterwards, temperature, convergence time, and efficiency is analyzed to determine the results.

The results indicate that although both neural networks can be used to solve this class of problems, there are drawbacks. However, those drawbacks are inconsequential and the benefits of using neural networks to efficiently solve these problems outweigh the consequences. Furthermore, neural networks can provide an efficient way to solve these problems while also consuming the least amount of power compared to other methods. Certain things should be addressed in future research on this specific topic such as complexity of the scheduling task and better resource optimization.

V. Conclusion

Neural Networks have become a key to being able to understand and analyze vast amounts of data and, as such, have become a solid candidate for scheduling Hard-Real-Time task sets. The nature of the Hopfield Network requires modification to its standard mapping for it to be able to overcome certain inherent issues, but upon modification, this network is a stellar performer in the space real time scheduling. By making simple changes to mapping structure, the Hopfield Network, when implemented via analog circuits, is able to generate valid solutions to advanced combinatorial problems (TSP and HRT Scheduling) in a matter of a few microseconds. It also has the potential to be implemented in hardware. This will further propel hardware technology to a new Golden Age. This would help get over the modern day bottleneck of memory and computing power and allow computers to become more powerful. This is a huge step in the optimization process for dealing with many combinatorial problems and will no doubt lead to further leaps and bound.

VI. References

Papers Cited

1. Cardeira, C., and Z. Mammeri. "Neural Networks for Multiprocessor Real-Time Scheduling." Proceedings Sixth Euromicro Workshop on Real-Time Systems, doi:10.1109/emwrts.1994.336864.
2. Guo, Zhishan, and Sanjoy K. Baruah. "A Neurodynamic Approach for Real-Time Scheduling via Maximizing Piecewise Linear Utility." IEEE Transactions on Neural Networks and Learning Systems, vol. 27, no. 2, 2016, pp. 238–248., doi:10.1109/tnnls.2015.2466612.
3. Silva, M.p., et al. "Solving Real-Time Scheduling Problems with Hopfield-Type Neural Networks." EUROMICRO 97. Proceedings of the 23rd EUROMICRO Conference: New Frontiers of Information Technology (Cat. No.97TB100167), doi:10.1109/eurmic.1997.617400.
4. Yueh-Min Huang, and Chen, Ruey-Maw. "Solving Multiprocessor Job Scheduling with Resource and Timing Constraints Using Neural Network." 2002 IEEE Region 10 Conference on Computers, Communications, Control and Power Engineering. TENCOM 02. Proceedings., doi:10.1109/tencon.2002.1181350.

Figures

1. By Glosser.ca - Own work, Derivative of File:Artificial neural network.svg, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=24913461
2. By Zawersh at the English Wikipedia, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=4107292
3. Author's Own
4. Guo, Baruah *"A Neurodynamic Approach for Real-Time Scheduling via Maximizing Piecewise Linear Utility"*
5. Guo, Baruah, *"A Neurodynamic Approach for Real-Time Scheduling via Maximizing Piecewise Linear Utility"*
6. Guo, Baruah *"A Neurodynamic Approach for Real-Time Scheduling via Maximizing Piecewise Linear Utility"*
7. Huang, Chen, *"Scheduling Multiprocessor Job with Resource and Timing Constraints Using Neural Networks"*

Christian Vartanian contributed to **Abstract, Getting over the Hump,** and **Hopfield Neural Networks**

Daniel Hureira contributed to **Further Implementations**, and **Conclusion**