Random Neural Networks with Hierarchical Committees for Improved Routing in Wireless Mesh Networks with Interference

Artur Rataj¹

Received: 22 July 2019 / Accepted: 2 September 2019 / Published online: 12 October 2019
© The Author(s) 2019

Abstract

We propose a hierarchical (nested) variant of a recurrent random neural network (RNN) with reinforced learning, introduced by Gelenbe. Each neuron (committee) in a top-level RNN represents a different bottom-level RNN (or sub-committee). The bottom-level RNNs choose the best routing and the top-level RNN chooses the currently best bottom-level RNN. Each of the bottom RNNs is trained in a different way. When they differ in their choice of the best path, several cognitive packets are routed according to the different decisions. In that case, a respective ACK packet trains individual bottom RNNs and not all bottom RNNs at once. An example presents an optimisation of a real-time routing in a dense mesh network of wireless sensors relaying small metering messages between each other, until the messages reach a common gateway. The network is experiencing a periodic electromagnetic interference. The hierarchical variant causes a small increase in the number of smart packets but allows a considerably better routing quality.

Keywords Random neural network · Hierarchical neural network · Cognitive packet routing · Real-time routing · Wireless sensor · Electromagnetic interference

Introduction

A hierarchical structure of committees, where a higher-level committee votes for the best lower-level committee, is universally found in complex organisations [22]. A recurrent random neural network by Gelenbe [10] contains already single-level committees. We extend that hierarchy to a two-level one and test its real-time efficacy in a wireless mesh network.

One of the frequent requirements in metering application concerns real-time constraints. For example, a packet with metering data must be delivered with at most a given mean latency, having at most a given variance. In particular, hard real-time constraints may be present, which invalidate all packets with expired data. On the other hand, sizes of packets transmitting such data are often small, if only events like

its very occurrence or a scalar temperature are communicated. This may cause that parameters like a maximum bandwidth of a network are not a problem. Conversely, metering networks like radio-frequency mesh systems often operate in difficult industrial conditions or over an extended area, which may make them particularly susceptible to failures or different external interference.

There are many types of wireless mesh networks in use [2, 7]. Different methods of obtaining an optimal or near-optimal routing [3] may alleviate various issues like uneven performance or interference [19]. In particular, a dynamic routing reconfigurable in real time can be provided by a cognitive packet routing (CPN) introduced by Gelenbe [6, 12, 23] with, e.g. energy-aware variants [17] which can be important in battery-based wireless nodes. In a CPN, a so-called smart packet (SP) is transmitted through a network along a path partially based on decisions of recurrent random neural networks (RNNs) (exploitation) and partially randomised (exploration). Each node contains a separate RNN for each destination of packets. For scalability, such an RNN usually decides only about the next hop, i.e. which output of the node to choose. An SP once arrives at its destination, returns as an ACK packet, which provides different QoS data, like latency. That data train the RNNs using

This article is part of the topical collection “Modelling methods in Computer Systems, Networks and Bioinformatics” guest edited by Erol Gelenbe.

✉ Artur Rataj
arturrataj@gmail.com

¹ IITiS PAN, Bałtycka 5, Gliwice, Poland

reinforcement learning [6]. There are concrete implementations of CPN in modern networks [1] and in performant hardware [4].

Reinforcement learning involves trial-and-error, where a balance exists between a usage of an accumulated knowledge and the need of acquiring a new knowledge, e.g. due to changing network conditions. This is known as a conflict between exploitation and exploration, or alternatively identification and control [15, 18]. CPN involves an exploitation/exploration scheme where particular variants of the same problems are evident: if exploratory decisions are rare, the routing will adapt slowly to new conditions. Otherwise, if SP packets are often rerouted because of the exploration, it will be hard to maintain multi-hop paths. For example, if a probability of leaving an N -hop path by an exploratory SP is p_x at each of the N nodes involved, then the probability of following exactly that path by an SP is obviously $p_f = (1 - p_x)^N$. For typical values of $N = 10$ for a moderately long path and a $p_x = 0.1$ to avoid trapping in local minima, we obtain $p_f \approx 0.35$. Thus, in this case most ACK packets will report on paths different from the original one, which may thus be forgotten, despite its aptness, due to extinction [14], a phenomenon which is obviously found in RNNs due to their training method.

This paper introduces a hierarchical (nested) variant of a recurrent RNN, where each neuron in a higher level (tier) RNN represents a different lower-tier RNN, as seen in the example in Fig. 1. The bottom RNNs choose the best routing, as usual in CPN, while the additional top-level RNN chooses the currently best bottom-level RNN. Each of the bottom RNNs is trained with a more or less different data, as these RNNs have different tendencies of learning from ACK packets having different levels of their “exploratory” past. When the bottom RNNs differ in their choice of the best path, new cognitive packets are routed according to the different decisions, yet still the original SP continues along the path given by the best bottom RNN (according to the top RNN). If a packet is not an effect of an unequivocal

decision of the bottom RNNs, then the respective following ACK packet trains only an individual bottom RNN which is responsible for the decision, and not all bottom RNNs at once. In turn, a packet sent in an exploratory direction by some router does not have an RNN responsible and, thus, it eventually trains bottom RNNs simultaneously in the router involved.

The paper is constructed as follows: in next section, we present a similar research. The proposed architecture is formally defined in “A Hierarchical RNN 3”. Section “Sensitivity to Exploration 4” shows how the training of bottom-layer RNNs is differentiated by regulating their sensitivity to the exploration. In “Inertial Cargo Routing 5”, we discuss an inertial routing of cargo (dumb) packets, which poses a substantial improvement in the studied mesh network. Section “A model of wireless sensor network 6” presents a model of a wireless sensor mesh network. The model is occasionally disturbed by a variable external electromagnetic interference, as described in “Scenario of external electromagnetic interference 7”. The hierarchical RNNs are tested in a case study in “Results 8”. Finally, there is a discussion in last section.

Similar Research

Employing nested hierarchy in an neural network is a common, if a very varied technique. A deep feedforward neural network can be seen as a set of layers processing data of different level of abstraction [16], just like in biological systems, e.g. in the visual cortex [21]. Different competition schemes can be used to choose the best solution. Poorly performing parts of a network can, e.g. be pruned [13] or an explicit voting scheme can be used [24], which is similar to the top RNN in our approach. Another, but related topic is an individualised learning of same-level modules in a hierarchical neural network. Logically, if we have a number of such competing modules, we might not want them to behave in exactly the same way. Sometimes a desired variation may appear thanks to the very randomisation of weight or connections [20]. We may also introduce the variation by making the modules learn differently, e.g. react in different time scales [9] or feed them with differently distorted data [8]. In the latter paper, the modules do not compete, but rather their output is averaged to form a response.

The author is not aware of any nested architecture specific to RNNs as presented by Gelenbe, but nevertheless there exist a training improvements of these networks with multicast SP [11], a solution with resemblance to a creation of multiple unicast SP in the method presented here. An event triggering such a behaviour is different, though: in [11], a node without knowledge of its surroundings floods them with broadcast SP. This is expensive and thus its usage is limited. In our case, the

Fig. 1 A schematic example of a two-tier hierarchical recurrent RNN. This network classifies between 3 classes (outcomes), and there are two competing committees (classifiers) that differ in some aspect. P_i are the potentials of higher level neurons, where i is an index of one of the competing bottom level classifiers; Q_j^i are the potentials of bottom level neurons, where j is one of the outcomes

multiple unicast SPs are created if, as discussed, the bottom RNNs diverge in their decision about a future path.

A Hierarchical RNN

Let there be an RNN r_p^i at a hierarchy level h_i , $i = 1, 2, \dots, H$. If it is a bottom-most RNN, i.e. at h_1 , then a decision represented by its neuron $\mathcal{N}_j^{i,p}$, $j = 1, 2, \dots, D$, is directly j . Obviously, if $\mathcal{N}_j^{i,p}$ is the winning neuron, i.e. one with the highest potential, that decision becomes also the decision of r_p^i . If instead, the RNN is located at h_k , $k > 1$, a decision represented by $\mathcal{N}_j^{k,p}$ is equal to the decision taken by one of the RNN at h_{k-1} which is “nested” by $\mathcal{N}_j^{k,p}$, as schematically depicted in Fig. 1.

Let a number of neurons in each of RNNs at h_k be C_k^n . Let there be only a single RNN at h_H , so that the whole hierarchy of RNNs takes only a single decision. A number of RNNs C_i^r at each layer are easy to compute

$$C_i^r = \prod_{k>i} C_k^n. \quad (1)$$

See that the constituents r_p^i , $p = 1, 2, \dots, C_i^r$, apart from the interpretation of their decisions, are plain RNNs as introduced in [11] and thus may undergo various improvements, like in [5, 23], independently of forming together a nested hierarchical classifier of two-level committees, the bottom ones should normally differ in some aspect of decision making in order to be not redundant.

Sensitivity to Exploration

As just said, we need a method of differentiation of training of the bottom RNNs. Different schemes can be thought of, e.g. a different training speed which would result in the bottom RNNs forming short- and long-term memories. Here, we will individualise the bottom RNNs by giving them different sensitivities to exploration, thus by setting individually the discussed exploitation/exploration balance.

Let each SP store where on its path it has been routed in an exploratory way. When returning as an ACK packet, it informs the router about the exact number of exploratory decisions E from that router to the SP's original destination. Thus, E can be understood as an explorativeness of the path taken by the SP, after it left the exact router. Let a probability, that a bottom RNN r_p^1 is trained by an ACK packet (does not ignore it) be

$$\Pr_p^e = \frac{1}{1 + \frac{p-1}{C_i^r-1} r_E E} \quad (2)$$

where r_E is a global constant of an exploratory rejection.

Inertial Cargo Routing

A modification which turned out to be important was an inertia of routing of normal cargo packets (also called dumb in CPN terms). The routing without inertia, i.e. if a decision of an RNN is realised directly also in the case of cargo packets, reflects possible transitory disturbances resulting from an ongoing rebuilt of the routing (e.g. when retraining the RNNs). The inertia, in turn, allows a slow transition from the routing so far to a new routing, hopefully already stabilised and robust. As the inertia does not affect SPs, it does not also affect the speed of training of RNNs. The inertia is realised in a straightforward way: when a cargo packet asks about a routing decision, then instead of choosing the winning neurons using their current potentials, a moving average is used instead:

$$Q_k^* = (1 - \beta)Q_{k-1}^* + \beta Q_k \quad (3)$$

where k is a training iteration of a specific RNN, Q_k is the current potential of a neuron (as experienced by SPs) and Q_k^* is the current inertial potential (as experienced by cargo packets). The coefficient β controls the update speed of Q_k^* and its effect on the routing quality will be studied in Sect. 8. Equation (3) is used in all RNNs in the hierarchy.

A Model of Wireless Sensor Network

The network has a grid topology as seen in Fig. 2. A router at each node is both a sensor and a wireless relay; boundary nodes can also be gateways. All sensors want periodically to submit a sensed value to one of the gateways. This is also a CPN network with slight modifications concerning the routing of SPs:

- if all neighbours around a node were already visited by a given SP, the packet fails and transforms into an ACK packet;
- if instead, an RNN decides about making an SP collide with its own path so far, the router is asked again; this loop always ends, as each router makes sometimes exploratory decisions.

A cost (weight) w of an edge is abstract and represents a difficulty in communication. One of the possible interpretations is a risk of retransmission due to noise, a potentially critical parameter in case of real-time constraints. A weight may also express real positions of the routers, which do not necessarily need to be placed along a grid. An additional cost would in that case also be an effect of distance to be travelled by a radio signal. Further, we place the routers on a regular grid, though.

Fig. 2 An example 8x8 wireless mesh, with only a single active gateway: **(a)** initial state, flat RNNs, **(b)** hierarchical RNNs with three neurons in the top layer (or networks in the bottom layer); as seen, nine of these internally diverges due to a rebuilt caused by a moderate external electromagnetic interference. Node size and colour depict its rate of creation of packets; gateways are numbered, though only one (marked with an arrow) is a destination of packets. Edge thickness depicts easiness of communication, i.e. is inversely proportional to the respective weight. Sizes of filled satellite circles around a node depict potentials in a respective RNN, where a potential of the win-

ning neuron is distinguished with green (lighter) colour. In the case of a hierarchical RNN, each of the bottom-level RNNs is depicted along a separate circle. A green (lighter) colour of these circles depicts a neuron with a maximum potential in the top level RNN. A narrowing of an edge with red (lighter) colour border represents the strength of a possible interference. See that in both **(a)** and **(b)**, the neural networks have been initialised with the winning neurons representing the same routing paths; thus, the visible differences in the winning neurons (or directions) are the result of a training process (colour figure online)

Scenario of External Electromagnetic Interference

Let separate sources of electromagnetic interference appear and disappear as a result of a Poisson process, where the respective rates are λ and μ . These are external sources outside the routers; thus, a placement of a source is arbitrary, yet within the region of the grid. The placement is also uniformly random. Once created, a source does not move.

The level of interference f with a router at a distance r from a source of an intensity s is given by the strength of an electromagnetic field around a point source:

$$f = \frac{s}{r^2}. \quad (4)$$

Thus, a near source may completely disable a router and large s may weaken communications in a wide region within the mesh (see an example in Fig. 2b). Let a unit of r be the distance of two nearest routers.

We assume non-directional antennas, thus f affects only reception, and not transmission quality at r . See that it makes the edge weights asymmetric in case of a non-zero interference. A base weight w is affected by f as follows:

$$w' = w + f \quad (5)$$

where w' is an effective weight caused by s . We assume a superposition of interference (which may not be completely true when, e.g. different sources operate on similar well-defined (low noise) frequencies, a distinguishable constructive or destructive addition of carrier waves may occur). Because of the superposition, (5) should be applied recursively for each source.

Results

Let us consider a 8×8 grid with edge weights as depicted in Fig. 2. We will use different number of RNNs in the bottom layer C_1^n to compare performances. The number of neurons in each of these bottom RNNs is equal in turn to the number of neighbours of a router. The constant weights w (called further for simplification latencies) have been drawn using a uniform PDF in the range $(0.2, 1)$. Any router creates SPs in a Poisson process at a constant rate drawn using a uniform PDF in the range $(0.04, 0.2)$. A failed SP adds to its latency a penalty of $F_{SP}^t = 1000$ when training and a penalty of $F_{SP}^q = 10$ when computing a mean latency for a given routing, i.e. a routing quality $\bar{L}$, a measure further used to estimate the performance. We use two

values as F_{SP}^t turned out to give good results and F_{SP}^q seems to be more realistic given the delays in the example grid. Let $\lambda = 8 \cdot 10^{-5}$, $\mu = 8.5 \cdot 10^{-5}$, which results in zero to few sources on average. The RNNs are initialised so that they realise an optimal routing. To counterpart a stochastic nature of CPN, we will further use different runs of simulation for very similar coefficient values, instead of multiple runs for the same values. This will roughly estimate at once a trend and a variation between the individual runs —see, e.g. Fig. 4, with less and more “noisy” regions for different β .

For visualisation and unless otherwise stated, we will use a concrete scenario of interference illustrated in Fig. 3, with a simulation time of $2 \cdot 10^5$ time units. Darker gray region shows a theoretically optimal routing quality $\bar{L}_{opt}$; lighter gray region shows a routing quality $\bar{L}_{init}$ assuming an initial configuration of RNNs (an optimal adaptation to a situation without the interference, as initially there was none). The latter region would thus represent a performance in the case when the RNNs do not learn at all and, thus, the name *static* in the figure.

Let us begin with an estimation of a good routing inertia represented by β , with non-hierarchical (flat) RNNs. Let a probability of an exploratory decision p_x be in this case typical for CPN and equal to 0.1. A diagram of a resulting latency overhead $\bar{L}^+$, i.e. a difference between an actual and an optimal routing, is shown in Fig. 4. Let us choose a default $\beta = 0.005$ where $\bar{L}$ is both low and far from the regions of a considerably worse performance. Let us use the value to compare a direct routing vs inertial one (again Fig. 3). We see that the inertia acts as a low-band filter,

Fig. 4 Routing quality overhead against inertia, flat RNNs, $p_x = 0.1$

reducing spikes in latency but not visibly adding any substantial delays when the interference abruptly changes.

See Fig. 5 for a comparison of the flat RNNs with hierarchical ones for different values of C_1^n and r_E . For more averaged results, the simulation time has been extended 10 times to that in Fig. 3 and we repeat the simulations for very close values of p_x . Due to the said creation of new SP if RNNs diverge in their decisions, there has been a moderate increase of the total number of SPs of about 9% when a hierarchical architecture has been employed. On the contrary, the latency overhead has been reduced from approximately 0.54 to 0.33 at $p_x \in (0.06, 0.12)$. The reduction amounts to about a 39% decrease in the latency overhead, at the cost of a small increase of the number of SPs. See that the region of optimal p_x coincides with our initial choice of $p_x = 0.1$ in Fig. 4.

We do not see a clear winner among the hierarchical architectures, although it can be said that for $C_1^n = 3$, $r_E = 0.3$ there still has been a substantial spike in the considered

Fig. 3 Routing latency against a variable electromagnetic interference, flat RNNs, $\beta = 0.995$

Fig. 5 Routing quality overhead for different architectures, against explorativeness, $p_x = 0.1$, $\beta = 0.005$, simulation time $2 \cdot 10^6$

region of p_x and that the lowest overhead of 0.023 has been achieved for $C_1^n = 3, r_E = 0.7$, vs 0.046 for the flat architecture.

Let us compare the performance visually as it changes with time. Fig. 6 shows the flat variant with several widely varying p_x . As visible in the diagram, a large ratio of exploratory SPs does not allow the RNNs to settle at an optimal point. Conversely, a small number of these packets produce a number of peaks despite the routing inertia, which may signalise a predictable problem with the speed of readjustment of neurons. The hierarchical RNN in comparison readjusts usually fast after the interference modifies network conditions.

Discussion

A hierarchy of different committees does not form the whole picture of the method. For example, an exploratory packet, when sent by chance in the direction of the winning bottom RNN in some router, may likely follow a successful path found in part by that winning RNN. Yet as it will eventually train all bottom RNNs in that router at once, a transfer of knowledge about that successful path from the winning RNN to the losing ones might occur in this case. A study and design of a well-balanced isolation and communication between the bottom RNNs, so that we keep their differentiation but also allow exchange,

Fig. 6 A temporal chart of an effect of different exploitation/exploration balance on the routing quality: a comparison with the hierarchical variant

may possibly be a way of a further improvement of the proposed solution.

Acknowledgements This research was funded by the H2020 SerIoT project.

Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

References

1. Adeel A, Larijani H, Ahmadinia A. Random neural network based cognitive-enodeb deployment in LTE uplink. In: Global communications conference (GLOBECOM), 2015 IEEE, pp. 1–7. IEEE. 2015;
2. Akyildiz IF, Wang X, Wang W. Wireless mesh networks: a survey. *Comput Netw.* 2005;47(4):445–87.
3. Alotaibi E, Mukherjee B. A survey on routing algorithms for wireless ad-hoc and mesh networks. *Comput Netw.* 2012;56(2):940–65.
4. Basterrech S, Janoušek J, Snášel V. A Study of Random Neural Network Performance for Supervised Learning Tasks in CUDA. In: Pan JS, Snasel V, Corchado E, Abraham A, Wang SL, editors. *Intelligent data analysis and its applications, Volume II. Advances in intelligent systems and computing*, vol. 298. Springer: Cham; 2014. pp. 459–468.
5. Basterrech S, Mohammed S, Rubino G, Soliman M. Levenberg–marquardt training algorithms for random neural networks. *Comput J.* 2009;54(1):125–35.
6. Basterrech S, Rubino G. A tutorial about random neural networks in supervised learning. *CoRR* **abs/1609.04846** (2016). [arXiv:abs/1609.04846](https://arxiv.org/abs/1609.04846).
7. Benyamina D, Hafid A, Gendreau M. Wireless mesh networks design—a survey. *IEEE Commun Surv. Tutor.* 2012;14(2):299–310.
8. Ciresan D.C, Meier U, Gambardella L.M, Schmidhuber J. Convolutional neural network committees for handwritten character classification. In: *Document analysis and recognition (ICDAR)*, 2011 international conference on, pp. 1135–1139. IEEE 2011;
9. El Hihi S, Bengio Y. Hierarchical recurrent neural networks for long-term dependencies. In: *Proceedings of the 8th international conference on neural information processing systems*. MIT Press: Cambridge; 1996. pp. 493–499
10. Gelenbe E. Random neural networks with negative and positive signals and product form solution. *Neural Comput.* 1989;1(4):502–10.
11. Gelenbe E, Lent R. Power-aware ad hoc cognitive packet networks. *Ad Hoc Networks.* 2004;2(3):205–16. <https://doi.org/10.1016/j.adhoc.2004.03.009>. <http://www.sciencedirect.com/science/article/pii/S1570870504000241>. Quality of service in ad hoc networks.
12. Gelenbe E, Lent R, Montuori A, Xu Z. Cognitive packet networks: Qos and performance. In: *Modeling, analysis and simulation of computer and telecommunications systems, 2002. MASCOTS 2002. Proceedings. 10th IEEE international symposium on*, pp. 3–9. IEEE. 2002.
13. Hassibi B, Stork DG. Second order derivatives for network pruning: optimal brain surgeon. In: *Advances in neural information processing systems 5, [NIPS Conference]*. Morgan Kaufmann Publishers Inc.: San Francisco;1993. pp. 164–71.
14. Hulse SH, Deese J, Egeth H. *The psychology of learning*. New York: McGraw-Hill; 1975.
15. Ishii S, Yoshida W, Yoshimoto J. Control of exploitation-exploration meta-parameter in reinforcement learning. *Neural Netw.* 2002;15(4–6):665–87.
16. LeCun Y, Bengio Y, Hinton G. Deep learning. *Nature.* 2015;521(7553):436.
17. Mahmoodi T. Energy-aware routing in the cognitive packet network. *Performance Eval.* 2011;68(4):338–46.
18. Narendra K.S, Thathachar M.A. *Learning automata: an introduction*. Courier Corporation. 2012;
19. Pathak PH, Dutta R. A survey of network design problems and joint design approaches in wireless mesh networks. *IEEE Commun Surv Tutor.* 2011;13(3):396–428.
20. Rataj A. Evolvability by mimicking common properties of a nervous system and computer software. *Fundam Inform.* 2014;131(2):253–78.
21. Riesenhuber M, Poggio T. Hierarchical models of object recognition in cortex. *Nature Neurosci.* 1999;2(11):1019.
22. Sah RK, Stiglitz JE. Committees, hierarchies and polyarchies. *Econ J.* 1988;98(391):451–70.
23. Sakellari G. The cognitive packet network: a survey. *Comput J.* 2009;53(3):268–79.
24. Wang J, He H, Cao Y, Xu J, Zhao D. A hierarchical neural network architecture for classification. In: *International symposium on neural networks*, pp. 37–46. Springer. 2012;

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.