

A Parametric Study of CPN's Convergence Process

Antoine Desmet and Erol Gelenbe

1 Introduction

The Cognitive Packet network [1–4] is a QOS-oriented packet routing protocol, which is decentralised, distributed and adaptive. The CPN concept emerged in 1999, and has been applied to several network types: energy-constrained sensor networks [5], integration with IP networks [6], building graphs [7], etc., and to solve different problems access control [8], attack defense mechanisms [9], congestion management [10], emergency management [11] and more. A comprehensive overview of the work on CPN can be found in [12].

CPN has the particularity of being based on search techniques [13, 14] using dedicated exploratory packets, the “Smart Packets” (SPs), which explore various paths and let CPN maintain a current knowledge of the network. SPs can be issued by any node, and their next direction is chosen independently by each node they visit. A variety of algorithms exist to guide incoming SPs, and of particular interest for this paper, Random Neural Networks (RNNs) [15, 16]. RNNs “learn” which areas of the network are most worth exploring and direct SPs there. RNNs are hosted on each node and learn through reinforcement learning, with feedback provided by SPs which backtrack once they have identified a valid path.

This constant network exploration allows CPN to respond to changes in network conditions with low latency, yet without conducting extensive or systematic network searches, thanks to the intelligence provided by the RNNs. The performance and overhead-efficiency of CPN with RNNs is subject to several parameters. Of interest in this paper, is the influence of the “drift parameter”, which controls the ratio of SPs forwarded according to the RNN, as opposed to randomly. To the best of our knowledge, all of the literature on CPN reports on trials of CPN for particular applications, but provide little to no information on how parameters are set [17, 18]. Our paper

A. Desmet (✉) · E. Gelenbe

Department of Electrical and Electronics Engineering, Intelligent Systems
and Networks Group, Imperial College, London, UK
e-mail: a.desmet10@imperial.ac.uk

E. Gelenbe

e-mail: e.gelenbe@imperial.ac.uk

aims at addressing the gaps in the literature regarding the role played by each CPN parameter on the algorithm’s performance.

In the following section, we present the CPN algorithm and RNNs in further detail. These concepts have been extensively covered in the literature, therefore we limit this introduction to the parts which are strictly relevant to our experiments. The third section analyses the effect of the drift parameter on CPN’s performance during the initial knowledge buildup phase (convergence), using a bench-test setup. Finally, we draw conclusions on the overall influence of each metric and propose directions for future work.

2 CPN and RNN Operation

CPN relies on a small but constant flow of “Smart Packets” (SP) dedicated to network condition monitoring and new route discovery. SPs are routed on a hop-by-hop basis, where each node visited *independently* decides of their next direction. While a variety of algorithms can be used to guide SPs, the aim is to focus the stream of SPs in the most worthwhile areas of the network, while limiting random or complete graph searches. To prevent “lost” SPs from wasting resources, they are also given an upper limit on the number of nodes they can visit. Once a SP reaches its intended destination, it backtracks along its original path (with loops removed) and shares the information gathered along the way with every node. By gathering information from returning SPs, every node is able to build a “source-routing” route table, which is used to guide regular payload-carrying packets.

2.1 SP Routing

Smart Packets are routed on a hop-by-hop basis independently by each node. A variety of algorithms exist to determine the SP’s next hop: they can be forwarded in randomly-chosen directions, and will perform a random walk of the network. The “Bang-Bang” [19] algorithm assumes an a priori knowledge of the network, which allows nodes to determine an “acceptable” range of route performance. If the known route falls within the “acceptable” range, SPs follow the best-known route; otherwise, the node promotes exploration by forwarding SPs at random. This approach is limited as it relies on a priori knowledge of the graph. Another approach is the Sensible routing policy [20] which forwards SPs probabilistically, based on the last path performance returned by each neighbouring node. The main challenge with this approach is to determine a mapping between path quality and corresponding SP forwarding probabilities. Approaches based on Genetic Algorithms have also been presented in the literature [21]. Another approach consists of running a Random Neural Network (RNN) [15, 16] in each node, and let it “learn” the areas most worth exploring and directs SPs there.

2.2 RNNs

Random Neural Networks [15, 16] are a form of neural networks inspired from bio-physical neural networks, where, instead of having neuron activity defined as binary or continuous variable, it is defined as a *potential*, and nodes send “spikes” at random intervals to one another. RNNs have been applied to various problems, including task assignment [22], video and image compressing [23], and more. Each CPN node on the network runs an RNN, and a neuron is associated to each outgoing link. The node forwards incoming SPs to the node associated with the neuron with the highest level of excitation. Therefore, the global aim is to “train” the RNN so that the most excited neurons correspond to the neighbour nodes, which are most worth visiting. Training is done through reinforcement learning, using the information provided by successful SPs backtracking along their original path. An in-depth presentation of the RNN model and training process can be found in [15, 16]. CPN nodes occasionally disregard the RNN’s advice and forward incoming SPs in a random direction to promote network exploration. This occurs when the RNN is in its initial state, before any feedback is provided by successful SPs. Past this initial phase, the CPN nodes also probabilistically forward SPs at random to prevent the RNNs from becoming overtrained (by sending the SPs on the same paths over and over). Forwarding a SP at random is referred as “drifting”, and we define the *drift parameter* as the probability that an incoming SP will be forwarded according to the RNN’s advice.

3 Experiments

In this section, we present a bench-test of the CPN algorithm, with an aim to reveal how the drift parameter influences CPN’s performance. Our bench-test focusses on CPN’s initial knowledge gathering phase: the “convergence” process. This experiment was conducted as part of a project where CPN was used as a routing algorithm for emergency building evacuations. The graph we use in this bench-test represents a three-storey building and consists of 240 nodes and 400 edges. Thus, CPN’s aim is to identify a path from each node towards the nearest “sink”: one of two exits located on the first floor. Our experimental protocol consists of letting all node send one SP (we refer this as a “batch” of SPs), and after each batch, we consult the routing table of each node: a score of $Q_P = 0$ is assigned if the node has not resolved a path yet. Otherwise, the score correspond to the ratio (expressed as percentage) of the shortest path’s length (found using Dijkstra’s algorithm) over the length of the path identified by CPN—so that the score is 100 % if CPN identifies the shortest path.

4 Results

Let us consider two trivial case-studies using the extreme values of the drift parameter to explore its expected influence on CPN’s performance:

DriftParameter = 0. In this configuration, the nodes systematically disregard the RNN’s advice and forward SPs at random. SPs effectively perform a random walk of the network, and thus any finite path has a non-zero probability of being visited through random walk. However, a path involving more nodes, or nodes with a higher degree is less likely to be visited.

DriftParameter = 1. Let us recall that a newly-initialised RNN which has not received any feedback has all its neurons in a “tie”. This tie is broken by choosing one direction at random, thus SPs explore the network at random while searching for a valid path. As soon as a SP discovers a valid path, all neurons along that path receive positive reinforcement and become the most excited neurons. Because of the drift parameter setting, subsequent SPs will not drift and will follow the path of the most excited neurons (which corresponds to the path of the first successful SP), and further reinforce them indefinitely. This means CPN will only resolve one single path per departure node, and there are no guarantees on its optimality, since it was discovered through a random walk and is never further optimised through random exploration.

From these two case studies, we can infer that a low drift parameter guarantees that, given time, the optimal path will eventually be found, however the process might be extremely slow since the knowledge gathered by the RNN is disregarded. On the other hand, higher drift parameter values ensure a solution will be reached rapidly as information gathered by previous SPs is systematically re-used to guide the subsequent ones. However, this initial solution may be sub-optimal, and further improvement may be slow or limited because of an over-training phenomenon: the same sub-optimal path is reinforced over and over.

4.1 Overall Route Quality

Figure 1 illustrates the path resolution process for three representative values of drift parameter. The top graph illustrates the “slow-but-steady” learning process associated with low drift parameter values (i.e. mostly random SP movement): over a quarter of the departure nodes have no valid path by the second SP batch. At the end of the experiment, some nodes still have no path resolved, yet it is clear that CPN continues making small but continuous improvements at each step, as the box-plot gradually shrinks towards 100%. On the other hand, the bottom graph associated to a high use of the RNNs exhibits the quickest initial path resolution process: CPN has resolved a path for most departure nodes after sending only one batch of SP. However, compared with the middle graph (drift parameter = 0.5), the median takes longer to reach 100% in the long-term, and the box plots remain wider at the end

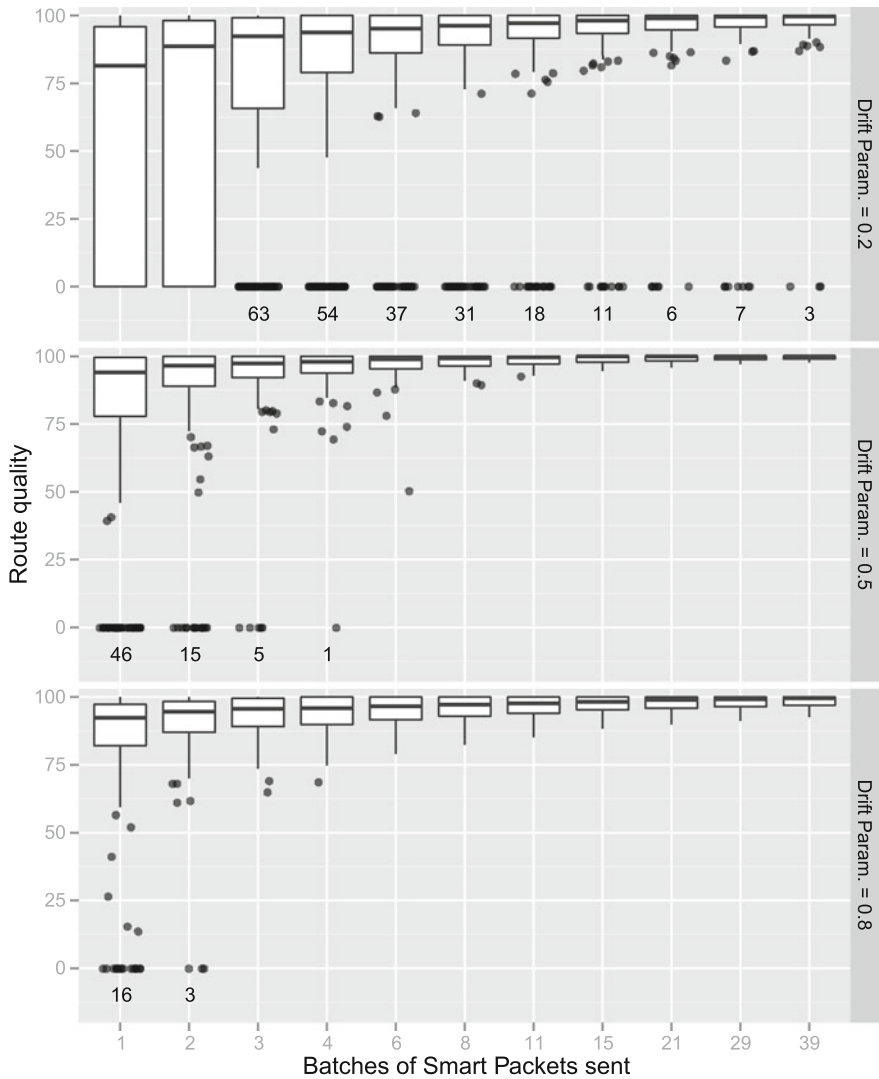


Fig. 1 Box-Plot showing the evolution in route quality based on the number of SPs sent. One sample for each 240 departure node. The *top* graph shows the “slow but steady” learning associated to low drift parameters, while the bottom one shows the “quick but approximate” discovery with high drift parameter

of the experiment. Past a fast learning phase, high drift parameter values lead to stagnation with sub-optimal values. Finally, the graph in the middle of Fig. 1 shows a “middle ground” where some of the initial resolution speed is “traded off” for a sustained higher improvement rate: while it takes five batches of SP for every node to acquire a path, the quality of the routes by the eighth SP batch is highest across

all experiments presented. This indicates that CPN’s behaviour can be modulated through the drift parameter, to meet the needs of applications which place a higher emphasis on either convergence speed, or solution quality.

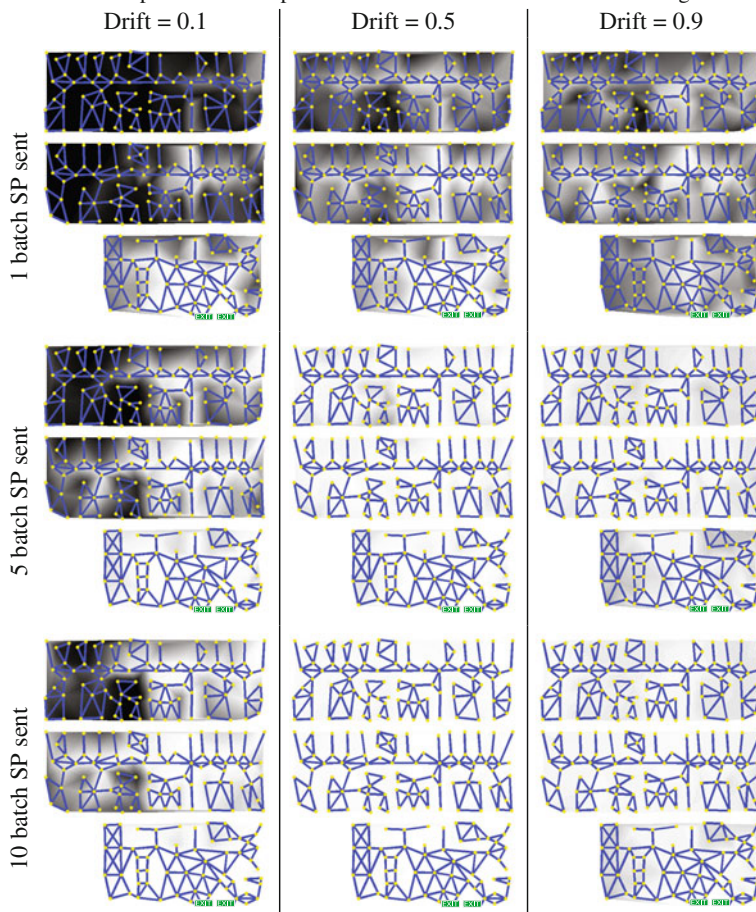
4.2 Spatial Convergence

Having considered the overall convergence process of CPN, we now focus on its spatial features, to see if the convergence rate depends on the node’s location. The figures arranged in Table 1 read as follows: vertically, each graph represents the path quality at three different stages of the convergence process: after $\{1, 5, 10\}$ SP batches. Horizontally, we display the graphs for three drift parameters values: $\{0.1, 0.5, 0.9\}$. Each cell contains a “flattened” view of the building graph, with the ground floor at the bottom. The graph is colorised in shades of grey where black corresponds to $Q_P = 0\%$ (no path) and white to $Q_P = 100\%$ (optimal path). Beyond corroborating our previous findings, this graph shows that CPN node in the following areas converge at a faster rate:

Around the sinks The likelihood of identifying a path during a random search is increased for shorter paths, and therefore, SPs starting near the sinks are at an advantage. This explains why route quality globally resembles a gradient function, where the quality decreases with distance from the sinks.

Main routes We recall that the information gathered by a SP is not only available to the node which issued it, but is also shared with every node visited along the way. Therefore, while leaf nodes can only rely on “their own” SPs to gather information on the network, nodes located near the graph’s backbone will harvest information from the many SPs transiting through them. This abundance of information helps them identifying the best path faster. The main corridors (horizontal edges across of the two upper floors) have a lighter shade than some other areas of the graph, which may be closer to the exits.

This experiment shows CPN rapidly establishes a “backbone” of high-quality paths, and then proceeds to explore more intricate areas. This ensures that packets will not have to travel too far before finding a node with a high-quality path to the sink. Our measurements show that CPN can be “tuned” to meet the needs of applications which can compromise on either solution quality, convergence speed or overhead. The analysis presented in this paper is still insufficient to make an informed decision on the optimal values of CPN’s parameters. An extension to this research would consist of running *dynamic* bench-tests, where we would modify some edge’s metric and measure the time taken by CPN to respond. We suspect that the drift parameter is likely to influence CPN’s dynamic latency, as CPN will have to conduct a wider search of the graph to detect changes in metrics and identify alternative paths: this will be largely controlled by the drift parameter. Finally, there are some parameters which we have not considered, such as the *hop count limit* or the *damping coefficient*, further experimentation should be conducted to determine their effect on CPN’s performance.

Table 1 Initial path resolution process across the three floors of the building

The lighter the colour the better the path found by CPN. The figure shows CPN starts by resolving a “backbone” of high-quality paths (main egress routes), and gradually progresses towards leaf nodes

References

1. E. Gelenbe, Cognitive packet network, U.S. Patent 6, 804,201 (2004)
2. E. Gelenbe, Z. Xu, E. Seref, Cognitive packet networks. In: 11th IEEE International Conference. Proceedings of Tools with Artificial Intelligence , pp 47–54 (1999)
3. E. Gelenbe, R. Lent, Z. Xu, Design and performance of cognitive packet networks. Perform. Eval. **46**(2), 155–176 (2001)
4. E. Gelenbe, R. Lent, Z. Xu, Towards Networks with Cognitive Packets, *Performance and QoS of Next Generation Networking* (Springer, London, 2001), pp. 3–17
5. L.A. Hey, Power aware smart routing in wireless sensor networks. In: Next Generation Internet Networks. NGI 2008, pp 195–202 (2008)
6. M. Gellman, Qos routing for real-time traffic. Ph D thesis, Electrical and Electronic Engineering Department, Imperial College London (2007)

7. H. Bi, A. Desmet, E. Gelenbe, *Routing Emergency Evacuees with Cognitive Packet Networks*, Lecture Notes in Electrical Engineering (Springer, Berlin, 2013)
8. E. Gelenbe, G. Sakellari, M. D'arienzo, Admission of qos aware users in a smart network. *ACM Trans. Auton. Adapt. Syst. (TAAS)* **3**(1), 4 (2008)
9. E. Gelenbe, M. Gellman, G. Loukas, An autonomic approach to denial of service defence. *World of Wireless Mobile and Multimedia Networks, WoWMoM 2005* in: Sixth IEEE International Symposium, pp. 537–541 (2005)
10. A. Desmet, E. Gelenbe, Reactive and proactive congestion management for emergency building evacuation. In: 38th Annual IEEE Conference on Local Computer Networks (LCN'13), Sydney, Australia, (2013)
11. A. Filippoupolitis, E. Gelenbe, A distributed decision support system for building evacuation. in: 2nd Conference on IEEE Human System Interactions. HSI'09. pp 323–330 (2009)
12. G. Sakellari, The cognitive packet network: a survey. *The Computer Journal* **53**(3), 268–279 (2010)
13. E. Gelenbe, Y. Cao, Autonomous search for mines. In: *AeroSense'97*, International Society for Optics and Photonics, pp 691–703 (1997)
14. E. Gelenbe, N. Schmajuk, J. Staddon, J. Reif, Autonomous search by robots and animals: A survey. *Robotics Auton. Syst.* **22**(1), 23–34 (1997)
15. E. Gelenbe, Random neural networks with negative and positive signals and product form solution. *Neural Computation* **1**(4), 502–510 (1989)
16. E. Gelenbe, Learning in the recurrent random neural network. *Neural Computation* **5**(1), 154–164 (1993)
17. E. Gelenbe, R. Lent, A. Nunez, Self-aware networks and QoS. *Proc. of the IEEE* **92**(9), 1478–1489 (2004)
18. G. Sakellari, E. Gelenbe, Adaptive resilience of the cognitive packet network in the presence of network worms. In: *Proceedings of the NATO Symposium on C3I for Crisis, Emergency and Consequence Management*, pp 11–12 (2009)
19. E. Gelenbe, E. Seref, Z. Xu, Simulation with learning agents. *Proc. IEEE* **89**(2), 148–157 (2001)
20. E. Gelenbe, Sensible decisions based on QoS. *Comput. Manag. Sci.* **1**(1), 1–14 (2003)
21. E. Gelenbe, P. Liu, J. Laine, Genetic algorithms for route discovery. *IEEE Trans. Syst., Man, Cybern., Part B: Cybern.* **36**(6), 1247–1254 (2006). doi:[10.1109/TSMCB.2006.873213](https://doi.org/10.1109/TSMCB.2006.873213)
22. Q. Han, Managing emergencies optimally using a random neural network-based algorithm. *Future Internet* **5**(4), 515–534 (2013)
23. C. Cramer, E. Gelenbe, P. Gelenbe, Image and video compression. *IEEE Potentials* **17**(1), 29–33 (1998)

<http://www.springer.com/978-3-319-09464-9>

Information Sciences and Systems 2014
Proceedings of the 29th International Symposium on
Computer and Information Sciences
Czachórski, T.; Gelenbe, E.; Lent, R. (Eds.)
2014, XVII, 423 p. 130 illus., 90 illus. in color.,
Hardcover
ISBN: 978-3-319-09464-9