QoS routing with incomplete information by analog computing algorithms

J. [1]LEVENDOVSZKY(⌧), A. [1]FANCSALI, CS. [1]VÉGSŐ, G. [2]RÉTVÁRI

Budapest University of Technology and Economics,

[1]Department of Telecommunications,

[2]Department of Telecommunications and Telematics

Pázmány Péter sétány 1/D., Budapest 1117, Hungary

Email: levendov@hit.bme.hu
Phone: +36 1 463 2485
Fax: +36 1 463 3263

Abstract

The paper proposes novel algorithms for Quality of Service (QoS) routing in IP networks. The new algorithms can handle incomplete information, when link measures (e.g. link delays, bandwidths ...etc.) are assumed to be random variables. Incomplete information can arise due to aggregated information in PNNI and OSPF routing protocols, which make link measures characterized by their corresponding p.d.f. It will be demonstrated that the task of QoS routing can be viewed as quadratic optimization. Therefore, neural based optimization algorithms implemented on an analog computer (CNN) can provide fast routing algorithms even in the case of incomplete information. As a result, real-time routing can be carried out to meet end-to-end QoS (such as end-to-end delay) requirements.

Keywords: QoS Routing Cellular Neural Networks Hopfield Networks

1. Introduction

One of the major endeavors in packet switched communication networking is to ensure QoS routing. This task boils down to select paths, which satisfy given end-to-end delay or bandwidth requirements [1,2,4,13]. As a result, QoS routing is perceived as an optimization problem to search over different quality paths and choose one for which the end-to-end QoS demands are met. Unfortunately, this problem, in general, cannot be reduced to the well-known shortest path routing (tractable by the Bellman-

Ford or Dijkstra algorithms in polynomial time [13]). Furthermore, if link QoS parameters (e.g. link delay or available bandwidth) are regarded as random variables then routing can become an NP hard problem [3]. In this case the aim is to select a path which guarantees the fulfillment end-to-end QoS criteria with maximum probability. We term this type of routing as a Maximum Likely Path Selection (MLPS) procedure. In this paper, MLPS will be reduced to a quadratic optimization, which can then be carried out by an analog computer called Cellular Neural Network (CNN).

The assumption that link QoS parameters are random variables is made on the following premises: (i) information aggregation (i.e., in the case of distant network components delay information is aggregated into an average delay) as detailed in OSPF and PNNI protocols [1,8], or (ii) randomly fluctuating delays or available bandwidths (where the current values of these parameters depend on the momentary traffic scenario [2]). In both cases, routing must be performed with incomplete information, which means that the routing algorithm is designed to select a path, which can fulfill end-to-end QoS criteria with maximal probability.

2. Routing with incomplete information

To model the routing problem let us assume that the following quantities are given:

- there is a graph $G(V,E)$ representing the network topology;
- each link $(u,v)\in E$ has some QoS descriptors $\delta_{(u,v)}$ (e.g. $w_{(u,v)}$ bandwidth or $\tau_{(u,v)}$ delay) which are assumed to be independent random variables subject to a probability distribution function $F_{(u,v)}(x)=P\left(\delta_{(u,v)}<x\right)$;
- there is an end-to-end QoS criterion (e.g. $\min_{(u,v)\in\mathrm{R}}\delta_{(u,v)}\geq A$ for some A in the case of bandwidth requirement or $\sum_{(u,v)\in\mathrm{R}}\delta_{(u,v)}<T$ in the case of end-to-end delay requirement), where R stands for a path connecting a predefined source node s and a destination node f;
- the objective is to find an optimal path $\tilde{\mathrm{R}}$ which most likely fulfills the given QoS criterion, namely:

$$\tilde{\mathrm{R}}:\max_{\mathrm{R}}P\left(\min_{(u,v)\in\mathrm{R}}\delta_{(u,v)}\geq A\right)\quad(1a)\qquad\text{or}\qquad\tilde{\mathrm{R}}:\max_{\mathrm{R}}P\left(\sum\nolimits_{(u,v)\in\mathrm{R}}\delta_{(u,v)}<T\right)\quad(1b)$$

One must note that in the first case δ is said to be a "bottleneck" type of link measure, whereas in the second case δ is said to be an additive type of link measure.

The path $\tilde{\mathsf{R}}$, introduced above, will be referred to as the Most Likely Path (MLP). It is well known that Shortest Path Routing (SPR) can be solved in polynomial complexity by the Dijkstra or Bellman-Ford algorithms. Therefore, mapping an MLP problem into an SPR is equivalent with proving that MLP can be solved in polynomial time. The following lemma establishes that a bottleneck measure MLP can easily be solved by using traditional SPR algorithms.

Lemma 1: The solution of $\tilde{\mathsf{R}}: \max_{\mathsf{R}} P\left(\min_{(u,v)\in\mathsf{R}} \delta_{(u,v)} \geq A\right)$ is equivalent to solving a traditional shortest path problem with the metric assigned to the $(u,v)th$ link being $\mu(u,v) = -\log P\left(\delta_{(u,v)} \geq A\right) = -\log\left\{1 - F_{(u,v)}(A)\right\}$.

Proof: We seek the path $\tilde{\mathsf{R}}: \max_{\mathsf{R}} P\left(\min_{(u,v)\in\mathsf{R}} \delta_{(u,v)} \geq A\right)$, which is equivalent to $\tilde{\mathsf{R}}$: $\max_{\mathsf{R}} P\left(\bigcap_{(u,v)\in\mathsf{R}} \delta_{(u,v)} \geq A\right)$. Assuming independent random variables $P\left(\bigcap_{(u,v)\in\mathsf{R}} \delta_{(u,v)} \geq A\right) =$ $\prod_{(u,v)\in\mathsf{R}} P\left(\delta_{(u,v)} \geq A\right)$, one can write $\tilde{\mathsf{R}}: \min_{\mathsf{R}} \sum_{(u,v)\in\mathsf{R}} -\log P\left(\delta_{(u,v)} \geq A\right)$. Therefore assigning measure μ as $\mu_{(u,v)} := -\log P\left(\delta_{(u,v)} \geq A\right)$ MLP routing can indeed be solved by SPR.

□

However, if the link descriptor is delay, then QoS routing yields an intractable problem, as stated by the following lemma.

Lemma 2: (Guérin et al) The solution of $\tilde{\mathsf{R}}: \max_{\mathsf{R}} P\left(\sum_{(u,v)\in\mathsf{R}} \delta_{(u,v)} < T\right)$ (which will be referred to as Delay Problem (DP)) in general is NP hard.

The proof is based on the fact that the problem of $\tilde{\mathsf{R}}: P\left(\sum_{(u,v)\in\mathsf{R}} \delta_{(u,v)} < T\right) > \pi$ (where $0 < \pi < 1$ is some given threshold) is also intractable. For further details regarding the proof, see [3].

Therefore, our effort is focused on introducing special constraints under which the optimization problem (1b) lends itself to analytical tractability, still preserving the main attributes of the problem (i.e., yielding results which are still relevant to practical networking scenarios).

3. MLPS as a quadratic optimization problem

In this section, we reduce MLPS to a quadratic optimization problem. In order to obtain a formal model let us introduce the following quantities:

- The link delay (or other QoS parameters) associated with link (u,v) is assumed to be a discrete random variable taking its values from a finite set $\tau_{(u,v)} \in \{T_1,...,T_M\} = \mathrm{T}$ with a discrete probability mass function $p_{(u,v),1},....., p_{(u,v),M}$. From this notation is clear that each link takes its delay value from the same set T, however, the probability mass function characterizing the random delay of link (u,v) can vary from link to link.
- These probability mass functions are summarized in the three-dimensional array $\overline{\overline{\overline{R}}} \to R_{ijn} = -\log P\left(\tau_{(i,j)} = T_n\right) \geq 0$. More precisely, R_{ijn} indicates the negative logarithm of the probability of taking a hop from node i to node j with delay T_n.
- The delay structure of the graph is summarized in the three-dimensional array: $\overline{\overline{\overline{D}}} \to D_{ijn} = T_n$, where the element D_{ijn} indicates that hopping from node i to node j introduces delay T_n in the path.
- A path together with its delay is represented by a three-dimensional array
$$\overline{\overline{\overline{V}}} \to V_{ijn} = \begin{cases} 1 & \text{if arriving at node } j \text{ at stage } i \text{ with delay } T_n \\ 0 & \text{otherwise} \end{cases}$$
- A Start Point Indicator Array (SPIA) and an End Point Indicator Array (EPIA) defined as
$$S_{ijn} = \begin{cases} 1 & \text{if } i=1, j=s \\ 0 & \text{otherwise} \end{cases} \qquad E_{ijn}^{(k)} = \begin{cases} 1 & \text{if } i=k+1, j=f \\ 0 & \text{otherwise} \end{cases}$$

  SPIA expresses that the path must start at node s, whereas EPIA indicates that the path ends at node f after k steps.

One must note that the three-dimensional arrays defined above have N^2M number of elements. In the forthcoming discussion we only consider the task of solving the problem of k-hop routing (finding a path containing only k links). This assumption is widely used to make the problem tractable.

3.1 Properties of a valid path

From the definitions given above, one can summarize the properties of a valid path as follows:

1. The two-dimensional projection V_{ijn} of a three-dimensional array $\overline{\overline{\overline{V}}}$ for a fixed n can have at most one element different from zero in each row (for any given n). This element equals 1.
2. The two-dimensional projection V_{ijn} of a three-dimensional array $\overline{\overline{\overline{V}}}$ for a fixed n can have at most one element different from zero in each column (for any given n). This element equals to 1.
3. The three-dimensional array $\overline{\overline{\overline{V}}}$ must have only $k+1$ elements equal to 1 and the remaining elements should be zero.
4. If the path should start from node s then at least one element is 1 in the "depth-row" of $\overline{\overline{\overline{V}}}$, namely V_{1sn} for some n, $n = 1,..., M$.
5. If the path should end at node f after k number hops (assuming k-hop routing), then at least one element is 1 in the "depth-row" of $\overline{\overline{\overline{V}}}$, namely $V_{k+1fn} = 1$ for some n, $n = 1,...., M$.

3.2 QoS routing as a constrained quadratic optimization problem

As could be seen in the last section, MLPS should take place over the space of three-dimensional data arrays which fulfill Criteria 1,..,5, respectively. This space is denoted by $\mathbf{V}$. Therefore, the objective is to select a path (or a corresponding $\overline{\overline{\overline{V}}}$), for which

$$\overline{\overline{\overline{V}}}_{opt} : \min_{\overline{\overline{\overline{V}}} \in \mathbf{V}} \sum_{i=1}^{k} \sum_{j=1}^{N} \sum_{l=1}^{N} \sum_{n=1}^{M} \sum_{m=1}^{M} V_{ijn} R_{jln} V_{i+1lm} + \left(\sum_{i=1}^{k} \sum_{j=1}^{N} \sum_{l=1}^{N} \sum_{n=1}^{M} \sum_{m=1}^{M} V_{ijn} D_{jln} V_{i+1lm} - T \right) \quad (2)$$

where T indicates the end-to-end delay requirement. One can easily see that the first term in the summation is related to the probability of a path as follows:

$$\sum_{i=1}^{k}\sum_{j=1}^{N}\sum_{l=1}^{N}\sum_{n=1}^{M}\sum_{m=1}^{M} V_{ijn}R_{jln}V_{i+1lm} = \sum_{(u,v)\in \mathsf{R}} -\log P(\tau_{(u,v)} = T_n)\Big|_{n:V_{uvn}=1} =$$
$$= -\log \prod_{(u,v)\in \mathsf{R}} P(\tau_{(u,v)} = T_n)\Big|_{n:V_{uvn}=1} = -\log P\left(\bigcup_{(u,v)\in \mathsf{R}} (\tau_{(u,v)} = T_n)\Big|_{n:V_{uvn}=1} \right) \tag{3}$$

To minimize (3) means that the array $\overline{\overline{\overline{V}}}$ is going to represent the most probable delay realization along the path R.

The second term $\sum_i \sum_j \sum_l \sum_n \sum_m V_{ijn}D_{jln}V_{i+1lm} - T$ expresses the QoS constraint enforced over the whole path, namely the following condition must hold: $\sum_i \sum_j \sum_l \sum_n \sum_m V_{ijn}D_{jln}V_{i+1lm} = \sum_{(u,v)\in \mathsf{R}} T_n\big|_{n:V_{uvn}=1} \leq A$ if the realization of the random sequence of the link delays in the path are $\tau_{(u,v)} = T_n\big|_{n:V_{uvn}=1}$ $\forall (u,v) \in \mathsf{R}$. Therefore, minimizing the first and the second term at the same time will yield a path, which fulfills the constraint in the best manner.

Hopfield type of neural algorithms can solve quadratic optimization problems in a polynomial time over the space of binary data structures. Thus, to utilize Hopfield or CNN-based methods, one has to build in additional constraints in the goal function, which enforce the solution obtained over the full binary space to be a valid path. This can be done by defining the optimization function as follows [5]:

$$\sum_{i=1}^{k}\sum_{j=1}^{N}\sum_{l=1}^{N}\sum_{n=1}^{M}\sum_{m=1}^{M} V_{ijn}R_{jln}V_{i+1lm} + \left(\sum_{i=1}^{k}\sum_{j=1}^{N}\sum_{l=1}^{N}\sum_{n=1}^{M}\sum_{m=1}^{M} V_{ijn}D_{jln}V_{i+1lm} - T \right) + \left(\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{n=1}^{M} V_{ijn}S_{ijn} - 1 \right)^2 +$$
$$+ \left(\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{n=1}^{M} V_{ijn}E_{ijn}^{(k)} - 1 \right)^2 + \sum_{n=1}^{M}\sum_{l=1}^{N}\sum_{\substack{j=1 \\ j\neq i}}^{N} V_{iln}V_{jln} + \sum_{n=1}^{M}\sum_{l=1}^{N}\sum_{\substack{j=1 \\ j\neq i}}^{N} V_{lin}V_{ljn} + \left(\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{n=1}^{M} V_{ijn} - k - 1 \right)^2 . \tag{4}$$

The third and the fourth term in the summation enforce that the path starts from node s and ends at node f. The fifth and sixth term impose the constraint of orthogonal rows and columns in the array $\mathbf{V}$. Finally, the seventh term is responsible to ensure a k-hop routing allowing only $k+1$ ones in the data array $\mathbf{V}$.

The three-dimensional data array $\overline{\overline{\overline{V}}}$ can be converted into a binary vector by applying the following transformation: $y_{((i-1)N+j-1)M+n} = 2(V_{ijn} - 0.5)$. With this transformation QoS routing is reduced to a quadratic optimization over $\{-1,1\}^{N^2M}$.

3.3 Neural based routing algorithms

Since the problem has been transformed into a binary quadratic optimization, it can be solved by a corresponding Hopfield type of neural network. The state transition rule is given by the following equation:

$$y_l(k+1) = \operatorname{sgn}\left(\sum_{j=1}^{N^2 M} W_{lj} y_j(k) - b_l \right), \quad (5)$$

where the weights W_{lj} and components b_l can be calculated by a simple algorithm given as follows. We know that every given quadratic expression $\phi(\mathbf{y})$ can be written in the following general form: $\phi(\mathbf{y}) = \frac{1}{2}\mathbf{y}^T \mathbf{W}\mathbf{y} - \mathbf{b}^T \mathbf{y} + \phi_0$.

One can substitute unit vectors in $\phi(\mathbf{y})$ as follows

$$\begin{aligned} &\phi(\mathbf{e}_i) = \frac{1}{2} w_{ii} - b_i + \phi_0, \quad \phi(-\mathbf{e}_i) = \frac{1}{2} w_{ii} + b_i + \phi_0 \quad (\forall i) \\ &\phi(\mathbf{e}_i + \mathbf{e}_j) = \frac{1}{2}\left(w_{ii} + w_{jj}\right) + w_{ij} - b_i - b_j + \phi_0 \quad (\text{if } w_{ij} = w_{ji} \ \forall i, j \neq i). \end{aligned} \quad (6)$$

Combining these equations we obtains

$$\begin{aligned} b_i &= \phi(-\mathbf{e}_i) - \phi(\mathbf{e}_i) \\ w_{ii} &= \phi(\mathbf{e}_i) + \phi(-\mathbf{e}_i) - 2\phi_0 \\ w_{ij} &= \phi(\mathbf{e}_i + \mathbf{e}_j) - \frac{1}{2}(w_{ii} + w_{jj}) + b_i + b_j - \phi_0 \end{aligned} \qquad \begin{aligned} &(i = 1, 2, \ldots, N^2 M) \\ &(j = 1, 2, \ldots, N^2 M; j \neq i) \end{aligned}$$

In this way, matrix **W** and vector **b** can easily be identified.
It is also noteworthy that the optimization performance of the Hopfield net can greatly be improved by adding a noise to (5), yielding

$$y_l(k+1) = \operatorname{sgn}\left(\sum_{l=1}^{N^2 M} W_{lj} y_j(k) - b_l + \nu(k) \right), \quad (7)$$

where $\nu(k)$ is a random variable subject to logistic distribution $p_\nu(x) = \left(1 + e^{-\alpha(k)x}\right)^{-1}$. It can be proven that the stationary distribution of the Markov chain generated by (7) yields the global maximum with maximal probability [7]. More precisely,

$$\mathbf{y}_{opt} = \arg\max_{\mathbf{y} \in \{-1,1\}^{N \cdot N \cdot M}} \pi(\mathbf{y}) = \arg\max_{\mathbf{y} \in \{-1,1\}^{N \cdot N \cdot M}} \mathbf{y}^T \mathbf{W}\mathbf{y} - 2\mathbf{b}^T \mathbf{y},$$

where $\pi(\mathbf{y})$ denotes the stationary distribution.

3.4 Solution by CNN

Since the problem has been transformed into a binary quadratic optimization it can be solved by a corresponding CNN. As has been pointed out in [6], the optimization can be carried out by the following differential equation:

$$\begin{aligned} \dot{x}_{ijn}(t) = -x_{ijn}(t) + (\alpha - 1)V_{ijn}(t) - \sum_{m=1}^{M}\sum_{l=1}^{N}(R_{jln} + D_{jln})V_{i+1,l,m} \\ - \sum_{m=1}^{M}\sum_{l=1}^{N}(R_{ljn} + D_{ljn})V_{i-1,l,m} - A\chi_{ijn}\left(\overset{\equiv}{V}\right) + b_{ijn} \end{aligned} \qquad (8)$$

where $\chi_{ijn}\left(\overset{\equiv}{V}\right) = \sum_{m=1}^{M}\sum_{\substack{l=1\\ l\neq i}}^{k+1} V_{ljm} + \sum_{m=1}^{M}\sum_{\substack{q=1\\ q\neq j}}^{N} V_{iqm} + \sum_{l=1}^{k+1}\sum_{\substack{m=1\\ m\neq n}}^{M} V_{ljm} + \sum_{l=1}^{k+1}\sum_{\substack{q=1\\ q\neq j}}^{N} V_{lqn} + \sum_{q=1}^{N}\sum_{\substack{m=1\\ m\neq n}}^{M} V_{iqm} + \sum_{q=1}^{N}\sum_{\substack{l=1\\ l\neq i}}^{M} V_{lkn}.$

The values of the constants A, α and b_{ijn} are also given in [6].

If we implemented the method on Hopfield type neural network, its recursion would stabilize in $O\left((n_{\text{nodes}} n_{\text{conn}})^2\right)$ step, where n_{nodes} denotes the number of neurons (now being N^2M) and n_{conn} represents the number of neurons which are connected maximally to one neuron (in our case it is NM multiplied by a constant). From this $O\left(N^7 M^4\right)$ complexity is obtained considering that we have to run the method for k=1,2,…,N–1. The result seems to be disappointing. However, taking into account that we can implement the method on an analog structure which is able to run in μs range, the high complexity is not an issue.

It is also noteworthy that the optimization performance of the CNN can be greatly improved by adding a noise (Wiener process) to (8) (for further details see [6]).

Unfortunately, the present CNN-technology does not allow to directly apply (8) in the case of larger than 6-7-node networks because of the relatively complex connection structure in which remote cells are also in connection. Even now investigations are conducted in order to apply the method on a 2D-CNN which has real template structure (only the neighboring cells are connected [9,10]). Currently 64x64-cell CNN is available which can allow us to implement routing in the case of usual network size. One must note that the proposed method is suitable for the case of small networks, because of the hierarchical routing [6].

4. Performance analysis

In this section the performances of the newly developed QoS routing algorithms are analyzed by extensive simulation. In order to compare the algorithms we introduced a performance measure for the paths starting from node s and ending at node f denoted by $\eta(G,s,f,T)$. This is defined as the ratio of the probability that the path found by a given algorithm satisfies the end-to-end delay requirement T to that of the path found by an exhaustive search, given as follows:

$$\eta(G,s,f,T) := \frac{P\left(\sum_{(u,v)\in \mathsf{R}_{\text{found by a given algorithm}}} \tau_{(u,v)} < T\right)}{P\left(\sum_{(u,v)\in \mathsf{R}_{\text{found by exhaustive search}}} \tau_{(u,v)} < T\right)}. \qquad (9)$$

However, one can calculate the average $\eta(G,s,f,T)$ for a whole graph (i.e., for each possible starting node s and for each possible ending node $f \in V$, yielding the following performance function $\eta(G(V,E),T)$:

$$\eta(G,T) := \frac{1}{|V|^2 - |V|} \sum_{s\in V} \sum_{f\in V, f\neq s} \eta(s,f,T) \ . \qquad (10)$$

Furthermore, one can calculate the average performance with respect to the end-to-end QoS criterion (T) given as follows

$$\eta(G) = \frac{1}{T_{\max}} \int_0^{T_{\max}} \eta(G,T)\,dT \ . \qquad (11)$$

and the average performance over a set of randomly generated graphs (G)

$$\eta(T) = \frac{1}{|\mathsf{G}|} \sum_{G\in \mathsf{G}} \eta(G,T) \, . \qquad (12)$$

In the case of a given graph this measure only depends on the value of the actual end-to-end QoS requirement. The closer this function approximates the value 1, the better the performance of the corresponding routing algorithm is.

The test graph on which the algorithms have been tested was fully connected containing 7 nodes and its topology is similar to a typical Local Exchange Network. Equidistant link scaling was used along the delay axis. The link delays were chosen as Bernoulli random variables the expected value of which fell into the middle of the interval in which the link delay is assumed to be contained.

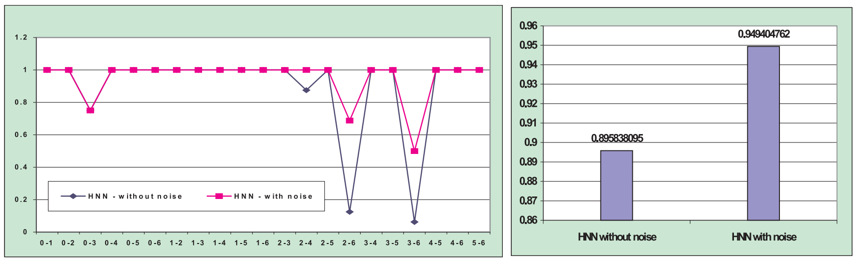

Figure 1. The performance of HNN methods changing the end points in the test graph

Figure 1 shows the performance of deterministic and noisy HNN routing algorithm on the test graph. On the horizontal axis the different source node - destination node pairs are indicated, whereas the values on the vertical axis correspond to the efficiency defined by (9). The corresponding bar chart indicates the related average performance defined by (10).

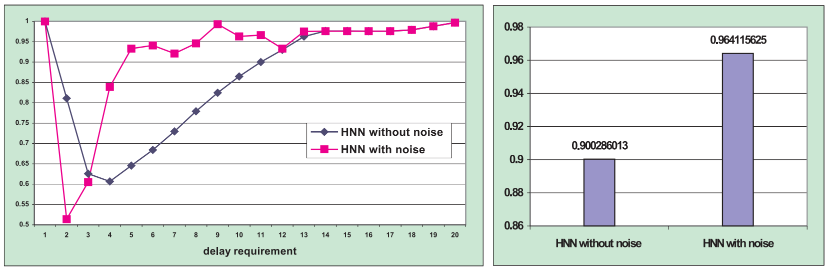

Figure 2. The performance of HNN methods changing the QoS requirement

Figure 2 exhibits how the efficiency depends on the QoS parameter (the overall delay *T*). On the horizontal axis the value of the QoS parameter indicated, whereas the values on the vertical axis correspond to the efficiency defined by (10). The corresponding bar chart indicates the related average performance defined by (11).

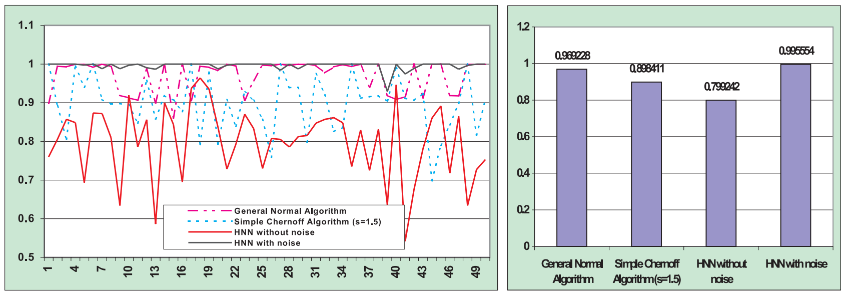

Figure 3. The performance of HNN methods generating graphs randomly

Figure 3 shows the performance of the deterministic and noisy Hopfield QoS routing algorithm. The figure also shows the performance of two other types of methods: “General Normal Algorithm” (using Gaussian approximation to reduce the original problem to SPR) and “Simple Chernoff Algorithm” (using large deviation approach to reduce the original problem to SPR). For further details regarding the algorithms, see [4]. The SPR is solved by Bellman-Ford algorithm in these two cases. The performance measure is taken over the ensemble of 50 randomly generated 7-node graphs. Each of them was generated with probability 0.8 to have connection between each node pair. While generating networks the non-separability was ensured. Table 1 shows the distributions of link values and QoS requirements [11], which were used to obtain the results shown in Figure 3.

Parameter	**Value**
T	Uniform distribution in $[30ms, 160ms]$.
$\delta_{(u,v)}$	Uniform distribution in $[0ms, 50ms]$.

Table 1. Simulation parameters

On the horizontal axis the networks are indicated, whereas the values on the vertical axis correspond to the efficiency defined by (10). The bar chart indicates the related average performance defined by (12). It is again verified by the figure, that the routing performance of the noisy HNN is rather high, it never goes below 0.9, even over a large ensemble of network topologies.

5. Conclusion

A novel CNN-based routing algorithm was introduced in the paper, which is able to meet end-to-end QoS requirements even in the case of incomplete information. Since routing was transformed into a binary quadratic optimization, CNN based solution became available, which yielded fast and a high performance routing. The quadratic cost function (4) easily can be modified to enable searching for paths with multiple cost functions. Unfortunately, the main advantages of recently developed algorithms [11,12,14]: working in distributed manner, searching for multiple paths are not be included in this CNN based solution. Moreover, the obtainable gain considering its running time (originated from analog operations) is restricted by the network size. Assuming large network topologies other types of approximations using digital computing techniques can have shorter running time.

Acknowledgment

The research reported here was supported by Analogical and Neurocomputing Laboratory, MTA SZTAKI. The authors are grateful for the comments and guidelines they received from Prof. Tamás Roska during their research.

References

[1] Cherukuri, R., Dykeman, D.: "PNNI draft specification", *ATM Forum* 94-0471, November 1995.

[2] Lorenz, D., Orda, A.: "QoS routing in networks with uncertain parameters", *IEEE/ACM Trans. Networking*, vol. 6., December 1998, pp. 768-778.

[3] Guérin, R., Orda, A.: "QoS routing in networks with inaccurate information: theory and algorithms", *IEEE/ACM Trans. Networking*, vol. 7., June 1999, pp. 350-364.

[4] Levendovszky, J., Rétvári, G., Dávid, T., Fancsali, A., Végső, Cs.: "QoS routing in packet switched networks - novel algorithms for routing with incomplete information", *Proceedings of 9th IFIP Working Conference on Performance Modelling and Evaluation of ATM & IP Networks*, Budapest, Hungary, pp. 249-260, 27-29 June, 2001.

[5] Levendovszky, J., Fancsali, A., Végső, Cs.: "CNN based algorithm for QoS routing with incomplete information", *22nd Symposium on Information Theory in Benelux*, Amsterdam, Netherlands, 15-16 May 2001.

[6] Levendovszky, J., Fancsali, A.: "Application of CNNs in modern communication technologies", Technical Report, Analogical and Neurocomputing Laboratory, MTA SZTAKI, DNS-7-2000.

[7] Jeney, G., Levendovszky, J.: "Stochastic Hopfield Network for Multi-user Detection", *European Conf. Wireless Techn.*, pp. 147-150, 2000.

[8] Lee, W.: "Spanning tree methods for link state aggregation in large communication networks", *Proc. INFOCOM*, Boston, MA, April 1995.

[9] Chua, L. O., Roska, T.: "The CNN Paradigm", *IEEE Trans. on Circuits and Systems I: Fundamental Theory and Applications, (CAS-I)*, Vol.40, No. 3, pp. 147-156, 1993.

[10] Chua, L. O. and Yang, L.: "Cellular Neural Networks: Theory", *IEEE Trans. on Circuits and Systems, (CAS)*, Vol.35. pp. 1257-1272, 1988.

[11] Chen, S., Nahrstedt, K.: "Distributed QoS Routing with Imprecise State Information", *Proceedings of 7th IEEE International Conference on Computer, Communications and Networks*, Lafayette, LA, pp. 614-621, October 1998.

[12] Chen, S., Nahrstedt, K.: "On Finding Multi-Constrained Paths", *Proceedings of 7th IEEE International Conference on Communications*, Atalanta, GA, pp. 874-879, June 1998.

[13] Chen, S., Nahrstedt, K.: "An overview of quality of service routing for next generation high-speed networks: Problems and solutions", *IEEE Network Magazine, Special Issue on Transmission and Distribution of Digital Video,* 12(6): 64-79, November-December 1998.

[14] Sun, Q., Langendorfer, H.: "A new distributed routing algorithm with end-to-end delay guarantee", *IWQoS'97*, May 1997.