IAES International Journal of Artificial Intelligence (IJ-AI)
Vol. 15, No. 6, December 2026, pp. 6015~6026
ISSN: 2252-8938, DOI: 10.11591/ijai.v15.i6.pp6015-6026

Evaluating value approximation and Q-routing for intelligent network on chip performance and efficiency

Smriti Srivastava, Minal Moharir, Prabu Jayant, Bandaru Jnyanadeep, Bipin Raj Channabasappa
Department of Computer Science and Engineering, RV College of Engineering, Bengaluru, India

Article Info

Article history:

Received Nov 3, 2025
Revised Aug 13, 2026
Accepted Sep 20, 2026

Keywords:

Neural networks
Packet routing
Q-routing
Reinforcement learning
Value approximation routing

ABSTRACT

This article investigates the performance of two reinforcement learning (RL)-based routing algorithms for packet routing: value approximation routing (VAR) and Q-routing. Performance is studied under uniform traffic patterns (transpose and bit-reverse) in terms of packet completion rate, latency, and energy. VAR uses a neural network to estimate value functions, handling large state spaces, whereas Q-routing uses the tabular form of the Q-learning, suitable for small networks. Simulation results show that VAR outperforms other schemes, with a 28.8% peak latency decrease (34.83 vs. 48.83 cycles) and a 63.2% energy consumption reduction (6,946.50 mJ vs. 18,892.50 mJ) achieved under the transpose traffic trace. For bit-reverse situations, VAR achieves 23% less worst-case latency (53.20 vs. 69.10 cycles) and consumes 5% less energy (17,230.50 mJ vs. 18,129.50 mJ). VAR displays more stable power profiles with dampened oscillations, suggesting that more advanced adaptation mechanisms work at lower injection rates (0.05 – 0.25). This paper reveals the capabilities for optimizing network performance and energy efficiency that can be achieved with an RL-based routing mechanism, shedding light on next-generation routing protocol design and machine learning-inspired communication networking in high-performance and energy-efficient networks.

This is an open access article under the CC BY-SA license.

Corresponding Author:

Smriti Srivastava
Department of Computer Science and Engineering, RV College of Engineering
Bengaluru, India
Email: smritis@rvce.edu.in

1. INTRODUCTION

Existing networks require adaptive routing schemes that can reduce latency, improve throughput, and optimize energy consumption in dynamic traffic scenarios [1]–[3]. Classical protocols fail during topology changes and varying loads [4], [5]. Artificial intelligence (AI) and routing capabilities have been investigated for some time because traditional routing protocols (static routing) cannot meet the requirement to accommodate changing conditions found in modern networks [6]–[8]. Packet routing can be researched as it relates to intelligent routing (i.e., dynamic networks) is expanding at a rapid rate with work being done in several areas, including networks on chip (NoC) architectures [1], [9]–[11], machine learning (ML) approaches [6], [8], [12], [13], and energy efficient protocols [3], [14].

Reinforcement learning (RL) based routing algorithms will utilize two (2) algorithmic approaches: value approximation routing (VAR) and Q-routing [14]–[16]. The VAR algorithm will utilize neural networks to develop estimates of value functions and derive generalizable (predictable) proposals for previously unobserved state transition [6], [13], [15]. The Q-routing algorithm is based on tabular Q-learning, where each state-action pair will be explicitly identified, defined and continuously modified [14], [16]. The Q-routing algorithm will provide, in particular, RL based routing algorithms with non-uniform traffic, given

that RL based algorithms have not yet shown an ability to perform well in such conditions [15]. Most RL based routing algorithms address primary objectives of energy efficiency at the expense of other similarly important objectives [3], [14]. Each of the proposed routing solutions will not be able to quickly respond to network dynamics or be scalable to larger sized networks; therefore, the network will be dealing with problems created by the use of routing solutions that cannot accommodate anticipated fluctuations [1], [6], [7]. Q-routing based routing solutions will not be viable given the high dimensionality of state-action spaces [6], [15], while VAR will not have been validated in real world applications when cost prohibitive resources are utilized for materialization of the routing solutions [17]–[19].

This research analyzes and evaluates the performance of VAR and Q-routing algorithms through simulation-based analysis [20]–[22]. The key performance indicators (KPIs) are evaluated: packet delivery ratio, mean delay, and energy consumption [23]–[25]. This paper address the problem of evaluating the performance, adaptability, and energy efficiency of VAR and Q-routing algorithms in various simulations. The main objective of this study is to evaluate the advantages and disadvantages of both algorithms. In such a way, the findings of this paper will assist network designers in determining when to apply ML-based routing solutions.

2. LITERATURE REVIEW

A study demonstrated that in distributed learning, nodes within a network make a routing decision based independently from the other nodes that form the network but together contribute to creating a global network optimization. Biglari *et al.* [1] surveyed architectures of NoC for heterogeneous multicore systems and indicated the challenges of dynamic communication patterns and adaptive routing mechanisms. Zhang *et al.* [6] provided an extensive survey of ML as it pertains to NoCs, including a review of performance modeling using ML, prediction using ML and techniques for optimizing the design of NoCs. Joshi and Ramasubramanian [4] suggested a hybrid crossbar ring (on-chip network) topology with the hope of improving multicore performance and showed that this topology provided improved throughput and reduced latency as compared to traditional mesh topologies. Li *et al.* [17] highlighted the high-bandwidth chiplet interconnects for advanced packaging technologies within the context of AI/ML applications and examined challenges associated with the design of on-chip interconnects and possible solutions through the use of feed-forward equalization to mitigate against degradation of signal integrity. Chatzopoulos *et al.* [20] introduced gem5-MARVEL, a microarchitecture-level resilience analysis framework for heterogeneous system-on-chip (SoC) architectures, providing consolidated fault injection infrastructure for reliability assessment. Kakoulli [7] presented a systematic review of latest innovations in intelligent NoC architectures, exploring ML and adaptive routing techniques for performance optimization. Nguyen *et al.* [18] developed a high-throughput and power-efficient field-programmable gate array (FPGA) implementation of YOLO CNN for object detection, demonstrating efficient neural network processing architectures. Feng *et al.* [2] proposed a low congestion-based routing algorithm for optical network-on-chip (ONoC) systems, achieving reduced network congestion and improved packet delivery rates.

Sun *et al.* [9] introduced metasurface optical network-on-chip (MSONoC), a metasurface-assisted hybrid waveguide / free-space optical (FSO) NoC architecture suitable for both wired and wireless communications, alleviating insertion loss and crosstalk noise. Liang *et al.* [12] developed non-dominated sorting whale optimization algorithm with genetic algorithm (NSWOAGA), a multi-objective non-dominated sorting whale optimization genetic algorithm for convolutional neural network-based on-chip networks, optimizing mapping strategies for improved performance. Zhao *et al.* [13] proposed a neural network-based traffic matrix prediction method incorporating inter-flow correlations for ONoC, with long short-term memory (LSTM) demonstrating superior performance compared to traditional prediction models. Yassin *et al.* [23] presented fast and accurate edge computing energy modeling and dynamic voltage and frequency scaling (DVFS) implementation in gem5 using system call emulation mode, enabling efficient power management simulation. Bossuet and Nino [24] explored power and frequency intrinsic channels on gem5, providing insights into emulating power-based characteristics in heterogeneous SoCs.

Reddy *et al.* [3] proposed techniques for enhancing reliability and energy efficiency in many-core processors through fault-tolerant NoC designs. Asadi [8] conducted a comprehensive study and holistic review of empowering NoC application mapping through ML techniques, surveying the state-of-the-art in ML-based NoC optimization. Binkert *et al.* [21] introduced the gem5 simulator, a merger of M5 and GEMS simulators providing highly configurable simulation framework, multiple ISAs, and diverse CPU models for architectural exploration. Power *et al.* [22] presented gem5-gpu, a heterogeneous CPU-GPU simulator that models tightly integrated systems with configurable memory hierarchies. Khan and Pasricha [15] proposed a RL framework with region-awareness and shared path experience for efficient routing in NoC, demonstrating significant performance improvements.

Zheng and Louri [14] explored energy-efficient NoC design using RL to automatically learn optimal control policies combining power-gating and DVFS techniques. Reza and Le [16] investigated RL enabled routing for high-performance NoC by proposing proactive routing techniques that work on optimal algorithms for congestion information. Asadi [10] presented a detailed analysis of optical router designs for ONoC architectures, examining microring resonators, Mach-Zehnder interferometers, and hybrid architectures. Werner *et al.* [11] surveyed ONoC architectures, discussing their strengths, weaknesses, and key enabling technologies for commercial adoption. Baharloo *et al.* [25] proposed a system-level reliability assessment framework for ONoCs, evaluating failure rates of microring resonators and optical routers. Romanov [5] developed routing algorithms for NoC based on ring circulant topologies, addressing communication subsystem optimization. Li *et al.* [19] presented a high-efficiency RL framework with hybrid architecture photonic integrated circuit for improving algorithm efficiency in materials synthesis tasks. However, many issues remain in NoC topologies and routing schemes. It is difficult to balance performance metrics and energy efficiency when most works focus on only one at a time. No studies directly compare neural-network-based approaches (e.g., VAR) with table-based approaches (e.g., Q-routing) under the same network and load conditions. The energy impact of RL-based routing decisions under different traffic loads also lacks in-depth study.

3. METHOD

This section comprises of technical implementation and efficiency scheme of Q-routing algorithm of packet routing by dint of dynamic networks. A learning mechanism adapts autonomously to varying network conditions utilizing distributed learning generated in network nodes. Q-routing is a significant improvement over network routing technology since it considers RL in packet transmission systems. This structure employs routers so they can acquire knowledge from experiences and modify activities to dynamic network conditions by enhancing packet distribution effectiveness and diminishing delays.

The Q-routing implementations are related to several critical parameters that must be properly configured. The exploration probability (epsilon) is 0.1, entailing routing choices by 10% being performed at random rather than by the learned optimal path. This averts failure into local solutions not necessarily on the optimal route. The discount factor (gamma) is 0.95 and shows how much the algorithm is aware of current actions versus future ones. A high value of the discount provides support for gains in the long term. The learning coefficient (alpha) shows how fast knowledge is coming into the decision-maker; this indicator was set to 0.1 to enable moderate changes of the Q-values. The SelectAction function uses ε-greedy policy to explore new ways of routing and utilize known successful routes. When a random number is not greater than ε, the router makes a choice among available next-hop routes randomly. Otherwise, it chooses the action with the highest Q-value showing the best result. If pure exploitation is sometimes leading to local minimums, pure exploration does not provide any useful outcome. The UpdateQTable function changes Q-values according to the routing results, rewarding the paths with the lowest latency, which leads to faster routing and better resource usage. The system determines the current router's position and uses learned behavior to choose next hops.

The VAR algorithm is constructed under the same premises of RL and applies changes in routing optimization tools. The exploration probability is 0.1, discount factor 0.95, and a lower learning coefficient 0.001 provides slower updates. In routing decisions, the SelectAction function preserves the ε-greedy approach incorporating state information for decisions. This enhances state awareness so that better alternatives can be chosen considering source-destination pairs, network load, and past traffic patterns. UPDATE focuses on refining value approximation (VA) and not Q-values. This algorithm is efficient on large networks where maintaining full Q-tables is infeasible. ROUTEPACKET in VAR has structural coherence with Q-routing yet includes optimization and destination checking loops to prevent repeated paths and failed transmissions.

In (1), Q (s, a) represents the Q-value of a state-action pair, which estimates the expected utility of taking action an in states. The learning rate α determines how quickly new information overrides old information, typically set between 0.1 and 0.3 for routing applications. Upon completion of an action, a reward (or immediate reward) is generated that will provide some indication of how well an action was performed, and will generally be inversely related to the latency of the link. The discount factor, (γ), will normally be about 0.95, which allows for the balancing of immediate reward with delayed reward. The overall approach is based on temporal difference learning (TD), whereby routers will progressively re-optimize their routing choices as feedback from network (or traffic) performance is implemented. A conceptual diagram of the value update process, which serves as the main backbone of the VAR, is also included. The VAR approach also provides a more streamlined method to Q-routing by being able to track the value of the states (i.e., not as a function of state-action pairs). This is a significant advantage because it

will result in a faster convergence to optimum values without sacrificing the level of performance that would be achieved under the original framework; i.e., (2).

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left(r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right) \quad (1)$$

$$V(s) \leftarrow V(s) + \alpha(r + \gamma V(s') - V(s)) \quad (2)$$

Value function $V(s)$, states represent an estimate of the desirability of the network traffic when routing packets to the destination. The rate at which convergence occurs is primarily determined by the learning rate (α), and as such, the value will tend to be much lower than that of Q-routing with respect to the linearity of convergence (i.e., typically less than or equal to 0.001). Reward (R) is the immediate feedback provided to the node (N) from the network, while γ represents how much the weight of the network will consider the next state versus the current state. The term s′ indicates the next state the router moves to after the current action. By focusing only on state values rather than state-action pairs, VAR decreases the dimensionality of the learning problem, which is advantageous in large-scale networks with many possible routes. Both Q-routing and VAR approaches must balance exploration and exploitation, often achieved using an ϵ-greedy strategy. This mechanism allows routers to discover new pathways while continuing to use established high-performing routes. The selection process follows the decision framework described in (3).

$$a = \{arg\ arg\ Q(s,a)\ ,\ with\ probability\ 1 - \epsilon\ random\ action \quad (3)$$

The mechanism of selection works according to the following principle: the router applies its knowledge that refers to the present moment and chooses the position with the highest value with 1–ε probability and explores other choices at random with mathematical ε probability. Usually, the initial ε value is taken equal to practically 0.1 and later decreased slowly as the router deeply studies the network environment. The above choice procedure prevents from making a premature choice of not very good routing paths since the possibility of exploration makes the routing algorithm discover more effective options. Algorithm 1 shows the Q-routing algorithm, while Algorithm 2 shows the VAR algorithm.

Algorithm 1. Q-routing algorithm for adaptive network routing

```
Require: Current router, destination, network state (topology and link latencies)
Ensure: Optimally routed packet
1:  procedure Q-ROUTING(packet, network_state)
2:      Initialize Q-table with zeros for all (router, destination, action) pairs
3:      ε ← 0.1                                          ▷ Exploration rate
4:      γ ← 0.95                                         ▷ Discount factor
5:      α ← 0.1                                          ▷ Learning rate
6:      function GETNEIGHBORS(router, network_state)
7:          return List of neighboring routers from network_state
8:      end function
9:      function RANDOMACTION(router, network_state)
10:         neighbors ← GETNEIGHBORS(router, network_state)
11:         if neighbors is empty then
12:             return ⊥                                 ▷ No valid actions; handle error
13:         end if
14:         return Random choice from neighbors
15:     end function
16:     function SELECTACTION(router, destination, network_state)
17:         if Random(0, 1) < ε then
18:             return RANDOMACTION(router, network_state)
19:         else
20:             neighbors ← GETNEIGHBORS(router, network_state)
21:             if neighbors is empty then
22:                 return ⊥                             ▷ No valid actions; handle error
23:             end if
24:             return arg max Q(router, destination, a)
                        a∈neighbors
25:         end if
26:     end function
27:     function UPDATEQTABLE(router, destination, action, reward, next_router)
28:         Q_current ← Q(router, destination, action)
29:         Q_max.next ← max Q(next_router, destination, a)
                         a∈GETNEIGHBORS(next_router, network_state)
30:         Q_target ← reward + γ · Q_max.next
31:         Q(router, destination, action) ← Q_current + α ·
                         (Q_target − Q_current)
```

```
32:     end function
33:     packet.path ← []                                  ▷ Initialize packet path
34:     while packet.current_location ≠ packet.destination do
35:         current_router ← packet.current_location
36:         action ← SELECTACTION(current_router, packet.destination,
                              network_state)
37:         if action = ⊥ then
38:              return ⊥                                   ▷ Unreachable destination; handle
error
39:         end if
40:         next_router ← action                           ▷ Action is the next router
41:         link_latency ← network_state.get_latency(current_router, next_router)
42:         reward ← −link_latency                         ▷ Negative latency as reward
43:         UPDATEQTABLE(current_router, packet.destination, action,
                        reward, next_router)
44:         packet.current_location ← next_router
45:         packet.path.append(next_router)
46:     end while
47:     return packet
48: end procedure
```

Algorithm 2. VAR algorithm

```
Require: Source router, destination router, network state
Ensure: Optimally routed packet with value approximation
1:  procedure VA-ROUTING(source_router, destination_router, network_state)
2:      ε ← 0.1                                           ▷ Exploration rate
3:      γ ← 0.95                                          ▷ Discount factor
4:      α ← 0.001                                         ▷ Learning rate
5:      function SELECTACTION(current_state)
6:          if Random(0, 1) < ε then
7:              return RandomAction()
8:          else
9:              return arg max V(current_state, a)
10:         end if
11:     end function
12:     function UPDATE(current_state, action, reward, next_state)
13:         V_current ← V(current_state)
14:         V_next ← V(next_state)
15:         V(current_state) ← V_current + α · (reward + γ · V_next − V_current)
16:     end function
17:     function ROUTEPACKET(packet, network)
18:         while packet.current_location ≠ packet.destination do
19:                              current_state  ←  GetNetworkState(packet.current_location,
packet.destination)
20:             action ← SELECTACTION(current_state)
21:             next_router ← ComputeNextRouter(packet.current_location, action)
22:             link_latency ← ComputeLinkLatency(packet.current_location, next_router)
23:             reward ← 1/(link_latency + 1)
24:             next_state ← GetNetworkState(next_router, packet.destination)
25:             UPDATE(current_state, action, reward, next_state)
26:             packet.current_location ← next_router
27:             packet.path.append(next_router)
28:         end while
29:         return packet
30:     end function
31: end procedure
```

This feature is useful in network scenarios where traffic, compute availability, or topology can vary with time. With exploration included, the router can still discover better routing strategies that would otherwise be concealed. Network throughput is one of the KPI's; this figure represents the actual amount of data successfully transmitted over a network (over a specific time period). The throughput number is thus a quantitative measure of whether the routing algorithm is able to effectively deliver data to its destination. The throughput number is calculated using the localized version of (4).

$$T = \frac{N_p}{t} \tag{4}$$

Where defines T, which is throughput measured by the number of packets delivered to their destination in a fixed unit of time (N, representing the total number of packets delivered during the period of observation). The increased value of throughput is a direct indication of better routing performance since it implies that the network can handle a larger amount of data. In RL-oriented routing protocols like Q-routing or VAR, the use of throughput as a KPI has become common and can be considered confirmation of the beneficial impact of

adaptive decision-making. Response time is the second crucial measurement parameter, which measures the delay when packet forwarding is performed through the infrastructure. A decline in response time indicates enhanced network responsiveness, which is particularly useful for network-sensitive applications. The average response time is computed using (5).

$$L = \frac{1}{N_p}\sum_{i=1}^{N_p} t_i \quad (5)$$

The average response time L goes through equation L = TI 1 + Ti 2 + Ti 3 + i + ki / Np. Every Ti is the accumulated time before a packet is received as the final destination after entering it in a network. Response time is often used as a component of the reward system in RL routing systems, with the shorter delay resulting in a larger reward. Such incentive mechanism inherently drives the system in the path that maximizes the latency. Delays especially when it comes to real-time communication, interactive games, and financial services as well as real-time interactivity services are also low response times, causing a slight delay that will go a long way in decreasing both usability and reliability. The other factor is an important consideration of power consumption which is vital at wireless sensor network, mobile platforms and other systems with constraints on energy. The performance of routing is also being assessed progressively with regards to power efficiency, and this is also being defined in (6).

$$E = \frac{T}{P} \quad (6)$$

Where E stands with the efficiency of power, T is the rate at which there is transmission, and P the sum of energy that is spent in watts. The larger E is, the more it has an energy-aware routing behavior, that is, a greater amount of data is provided with a power unit. Using reinforcement-based learning routing algorithm, energy awareness in the forms of a reward-function can be integrated, and hence routing algorithms can be deployed to promote using routes that would provide a balance between high throughput and low-power consumption. This type of optimization is particularly essential in the case of portable devices, and in the scenario of sustainable computing, the extension of operational lifetime and reduced environmental impact is critical. Success rate of delivery is provided in (7).

$$P_d = 1 - \prod_{i=1}^{h}(1 - P_i) \quad (7)$$

In (7) calculates overall delivery as $P_d = P_0 \times P_1 \times P_2$, where P_f equals the likelihood of success of link $i$and $h$is the number of hops in the route. This expression represents the cumulative effect of link failures along a path. RL routing algorithms can make routing choices that favor reliable delivery routes by incorporating delivery reliability in state representations or through reward functions. This is particularly important in areas where data integrity is critical, such as financial networks or maintenance and sensor systems. Deep learning value estimation (DLVE) is a derivative of the VAR method, which uses neural networks to estimate routing values in irregular and large topologies. It is suitable for high-dimensional state spaces in which standard table methods are impractical. The estimation process is expressed in (8).

$$V(s;\ \theta) = f(Wx\ +\ b) \quad (8)$$

In this equation, $V(s_{stat})$ represents the value functions of the estimated neural network, s is the state, and θ the neural network parameters such as weight matrices and biases. The transformation matrix W maps the input feature x, which defines the system state, into a latent representation space. The bias term b increases the ability of the model to fit data trends by increasing the number of degrees of freedom. To achieve the desired nonlinearity for learning complex network effects, the activation function is defined by the rectified linear unit (ReLU) to avoid the vanishing gradient issue. This topological structure allows for routing decisions not based on current topological metrics to be made based on learned representations resulting from prior experience. This increases the network's flexibility through the generalization of prior routing decisions to novel states of the network, thus allowing the network to adjust to new operational domains. Generalization is accomplished through the creation of learned representations which accurately represent significant aspects of the network's topology and the traffic patterns of the network. The bias term b serves an important purpose because it allows neurons to remain “on” (active) when the weighted addition of the input signals is close to zero.

All experiments used the cycle-accurate, event-driven gem5 simulator (gem5), which simulates the operation of each processing core of the processors. In each experiment, both types of reinforcement agents are initialized, and the hyper-parameters of both are subsequently modified. Traffic patterns were created

using the transpose permutation and the bit-reverse permutation. All packets were routed through an eight-node, two-dimensional mesh network using the appropriate routing policy for each node, which involved either updating the Q-table (Q-routing) or calculating the performance viability of the state-action pair using the value estimate derived from the value table (VAR). The data collected from each experiment were the performance metrics of the packets, the average end-to-end latency and the average power consumption. The initial exploratory rate (ε) was set at a value of 0.1 or a 10% probability of choosing randomly from among the available path choices. The initial discount factor (γ) value of 0.95 emphasizes that the agent's long-term reward will be weighted higher than the agent's short-term reward.

Learning rates were specified for each agent-type and were set to 0.1 for the Q-routing agent and 0.001 for the VA-routing agent. Two standard traffic patterns, transpose and bit-reverse, distributed packets in a complex manner that challenged routing decisions. Injection rates varied from 0.05 to 0.85, resulting in light to heavy network traffic. Several packets were sent until all reached their receivers or were discarded because of queue overflow. An ε-greedy policy was used for routing decisions. The SelectAction function selected the next hop according to the current network status, and the UPDATE function updated Q-values in Q-routing or neural network weights in VAR. Rewards were the inverse of packet delays. GEM5 collected metrics such as latency, throughput, power, and energy. Data were acquired at each injection rate, and results of multiple test runs were analyzed to minimize random differences. Python scripts processed the results and produced comparison charts.

4. RESULTS AND DISCUSSION

Routing algorithms are being tested under difficult conditions using bit-reverse and transpose traffic patterns [1], [2], [5]. VAR and Q-routing each handle different metrics (latency, power consumption, and packet delivery rates) when they compete for network resources [6], [7], [14]–[16]. The flowchart of both algorithms is shown in Figure 1.

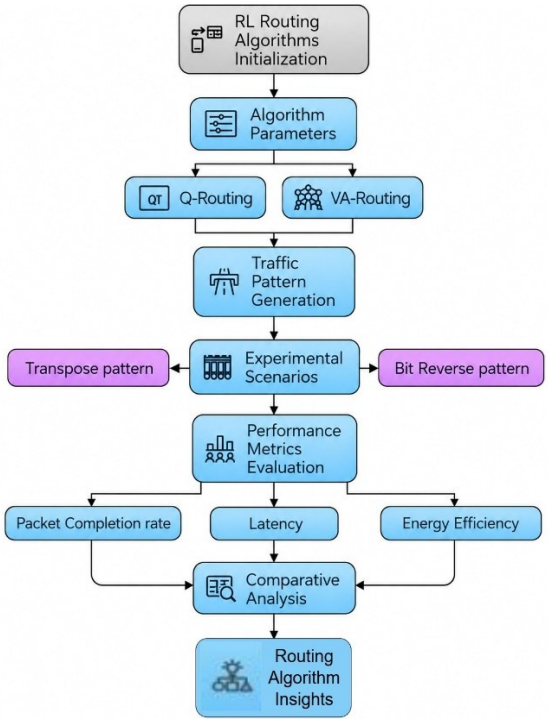
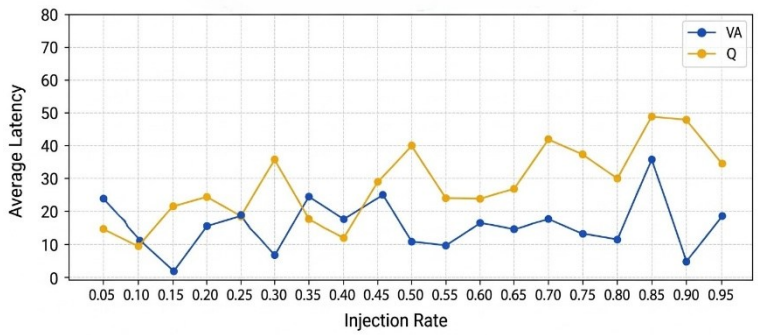
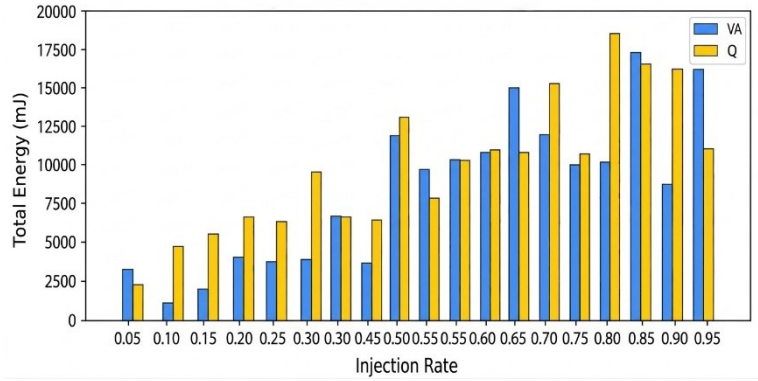


Figure 1. Implementation flowchart of Q-routing and VAR algorithms

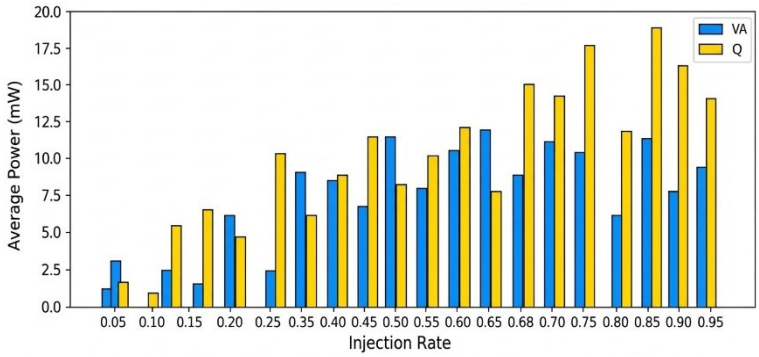
Performance is illustrated in Figure 2 for the transpose traffic patterns. The average packet latencies are shown in Figure 2(a) (injection rate vs. average packet latency), total energy consumption is shown in Figure 2(b), and average power consumption is shown in Figure 2(c) (transpose pattern). The VAR algorithm has been able to maintain a relatively flat latency profile range of 3.33:22 on injection rates from 0.05 to 0.25, which indicates its ability to handle an application with limited performance degradation due to initial network congestion. At an injection rate of 0.85 the VA algorithm shows a latency of 34.83 cycles, indicating that the algorithm has limited capabilities of performing congestion control.



(a)



(b)



(c)

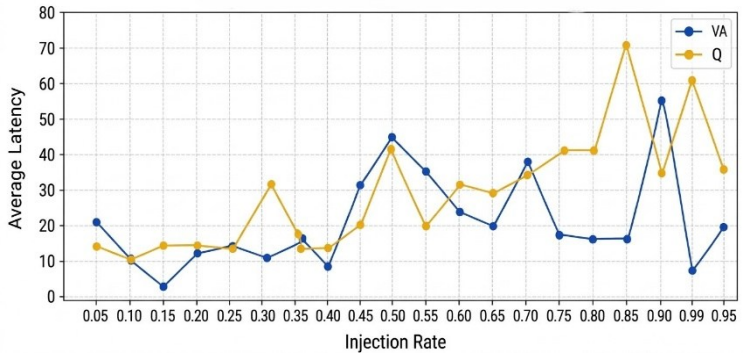
Figure 2. Performance metrics under transpose traffic pattern (a) injection rate vs. average packet latency, (b) injection rate vs. total energy consumption, and (c) injection rate vs. average power

In contrast, the Q-routing algorithm has experienced very high surges in latency and unexpected jumps when tested at moderate to high injection rates (peaks at: 0.70, and 0.90), indicating that Q-routing will not be able to handle complex transpose traffic patterns effectively. Additionally, Figure 2(b) illustrates the energy consumption of both algorithms operating under transpose traffic. Energy consumption for VAR ranges from 1497.00 mJ to 6946.50 mJ which demonstrates the VAR algorithm has superior management capabilities. The power profiles for VAR in Figure 2(c) indicate stable ranges (low load) from 1.50:6.95 mW, whereas Q-routing exhibits unstable spikes (17.52 mW @ 0.75, 18.89 mW @ 0.90). Thus, VAR should be the preferred algorithm due to its balanced trade-off between performance and power efficiencies compared to the simpler energy management strategies utilized by Q-routing [3], [14].

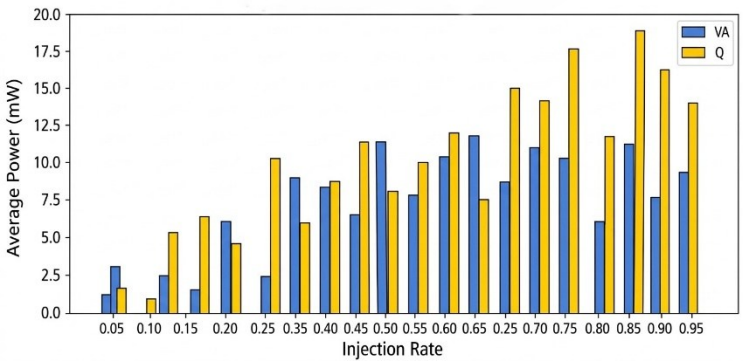
The difference in the energy behavior of the two algorithms is more evident as the injection rates increase. Unlike in the case of consumption of Q, the VA is more controlled and smoother and this attributes to adaptability. The VA algorithm is shown to be able to balance between performance and efficiency goals and energy management making it reliable under wide conditions of the network. At small injection rates in the packets, the VA algorithm does reduce energy consumption by applying dynamically adjusted resource-allocation strategies that match the energy consumption to its current network demand.

In comparison, the Q-learning scheme generates unreliable energy spikes: at medium and high loads, the power draw will suddenly increase without relying on the network demands. Both these spikes of 17.52 mW at injection rate 0.75 and 18.89 mW at injection rate 0.90 demonstrate limitations of Q-learning. At the light load conditions (0.05 - 0.20 injection rate), VA will maintain the minimum consumption 1.50 to 6.95 mW, which saves power without adversely affecting the operation. This energy consumption anomaly implies that the scalability of resources and congestion measures are to be enhanced, which questions the applicability of Q algorithm to the energy-constrained environments. Such attitude implies that the VA algorithm [16] remained stable in the realistic conditions of the network traffic and therefore, it is closer to the practical solutions in the context of the next generation networks in which the balance between the performance of the traffic and the power consumption have to be considered.

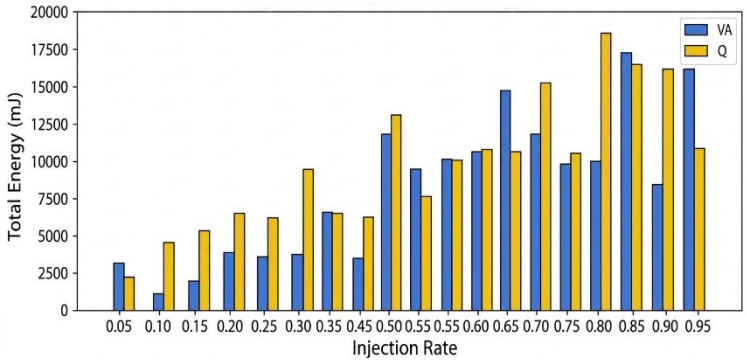
The bit-reverse pattern is used to characterize one of the hardest problems of adaptive routing schemes that are built by reversing the binary form of source address and making each node to address destination address dictated by this mathematical law. Figure 3 presents bit-reverse traffic results, where Figure 3(a) shows injection rate vs. average packet latency, Figure 3(b) shows average power consumption, and Figure 3(c) shows total energy consumption.



(a)



(b)



(c)

Figure 3. Bit-reverse traffic results (a) injection rate vs average packet latency, (b) injection rate vs average power, and (c) injection rate vs total energy

In Figure 3(a), VAR and Q-routing display complex behavioral differences under bit-reverse traffic. VAR exhibits latency peaks of 53.20, 39.94, and 31.17 cycles at injection rates 0.45 and 0.65. Q-routing shows maximum spikes at 0.8, with secondary peaks at 0.9 (59.99 cycles) and 0.7 (35.17 cycles), while remaining more stable at lower rates (7.80 – 12.05 cycles). Figure 3(b) shows low power usage (1.07 – 14.60 mW) in the 0.05 – 0.3 range for both algorithms. VAR experiences sharp rises to 17.23 mW (0.85) and 15.58 mW (0.65), while Q-routing displays irregular jumps peaking at 18.13 mW (0.8) and 15.67 mW (0.7). Figure 3(c) reveals low energy consumption (1,069.00 – 4,599.50 mJ) at lower rates. VAR reaches 17,230.50 mJ (0.85) and 15,580.00 mJ (0.65), while Q-routing shows erratic fluctuations: 18,129.50 mJ (0.8), 15,666.50 mJ (0.7), and 16,594.50 mJ (0.85).

5. CONCLUSION

A detailed comparative analysis of VAR and Q-routing is provided in this paper about transpose traffic pattern and bit-reverse traffic pattern in terms of latency, throughput, power consumption and energy efficiency. The VA scheme, which is founded on neural VA, competes favorably by latency control as well as high energy savings with respect to injection rate variation. Q-routing on the contrary although effective in simpler settings exhibited volatile power consumption and higher latency to heavy traffic loads. These findings demonstrate the usefulness of the function approximation of dynamic networks especially when large-scale systems with adaptive and energy-conscious routing are required. Further development will entail the investigation of a hybrid routing scheme that will unite the scalability of neural models with the convergence rate of the tabular ones. The implementation in real-world infrastructures, e.g., the internet of thing networks, 5G/6G system, and edge-cloud environment, will challenge the versatility and efficiency of these models even further. Moreover, live learning, predictive traffic modelling and combination with security conscience routing protocols are all possible directions of improving resilience and intelligence in the network of packet switching.

FUNDING INFORMATION

Authors state no funding involved.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (Credit) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Smriti Srivastava	✓							✓		✓		✓	✓	
Minal Moharir	✓	✓				✓		✓		✓	✓	✓	✓	✓
Prabu Jayant		✓	✓	✓	✓	✓	✓		✓	✓	✓			
Bandaru Jnyanadeep		✓	✓	✓	✓	✓		✓	✓	✓	✓			
Bipin Raj Channabasappa		✓			✓	✓	✓			✓				

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject administration
Fu : **Fu**nding acquisition

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding author, [SS], upon reasonable request.

REFERENCES

[1] S. Biglari, F. Hosseini, A. Upadhyay, and H. Zhao, "Survey of network-on-chip (NoC) for heterogeneous multicore systems," in *2024 IEEE 17th International Symposium on Embedded Multicore/Many-core Systems-on-Chip*, Dec. 2024, pp. 155–162, doi: 10.1109/MCSoC64144.2024.00036.

[2] J. Feng, D. Meng, Q. Yao, H. Yang, and J. Zhang, "Low congestion-based routing algorithm in optical network-on-chip," in *2024 22nd International Conference on Optical Communications and Networks (ICOCN)*, IEEE, Jul. 2024, pp. 1–3, doi: 10.1109/ICOCN63276.2024.10648518.
[3] B. N. K. Reddy, M. Z. U. Rahman, and A. L.-Ekuakille, "Enhancing reliability and energy efficiency in many-core processors through fault-tolerant network-on-chip," *IEEE Transactions on Network and Service Management*, vol. 21, no. 5, pp. 5049–5062, Oct. 2024, doi: 10.1109/TNSM.2024.3394886.
[4] A. D. Joshi and N. Ramasubramanian, "A hybrid crossbar-ring on chip network topology for performance improvement of multicore architectures," *International Journal of Information Technology*, vol. 15, no. 7, pp. 3967–3977, Oct. 2023, doi: 10.1007/s41870-023-01433-1.
[5] A. Y. Romanov, "Development of routing algorithms in networks-on-chip based on ring circulant topologies," *Heliyon*, vol. 5, no. 4, Apr. 2019, doi: 10.1016/j.heliyon.2019.e01516.
[6] X. Zhang, D. Dong, C. Li, S. Wang, and L. Xiao, "A survey of machine learning for network-on-chips," *Journal of Parallel and Distributed Computing*, vol. 186, Apr. 2024, doi: 10.1016/j.jpdc.2023.104778.
[7] E. Kakoulli, "Latest innovations in intelligent network-on-chip architectures: a systematic review," in *2024 17th IEEE/ACM International Workshop on Network on Chip Architectures*, 2024, pp. 1–6, doi: 10.1109/NoCArc64615.2024.10749933.
[8] Y. Asadi, "A comprehensive study and holistic review of empowering network-on-chip application mapping through machine learning techniques," *Discover Electronics*, vol. 1, no. 1, Oct. 2024, doi: 10.1007/s44291-024-00027-w.
[9] W. Sun *et al.*, "MSONoC: a metasurface-assisted hybrid waveguide/FSO network-on-chip architecture," *IEEE Transactions on Green Communications and Networking*, vol. 9, no. 3, pp. 873–888, Sep. 2025, doi: 10.1109/TGCN.2024.3506752.
[10] Y. Asadi, "Optical network-on-chip (ONoC) architectures: a detailed analysis of optical router designs," *Journal of Semiconductors*, vol. 46, no. 3, Mar. 2025, doi: 10.1088/1674-4926/24060006.
[11] S. Werner, J. Navaridas, and M. Luján, "A survey on optical network-on-chip architectures," *ACM Computing Surveys*, vol. 50, no. 6, pp. 1–37, Nov. 2018, doi: 10.1145/3131346.
[12] Z. Liang, F. Hu, B. Xu, and C. Wei, "Multi objective non dominated sorting whale optimization genetic algorithm for convolutional neural network-based on-chip networks," in *2024 5th International Seminar on Artificial Intelligence, Networking and Information Technology*, Mar. 2024, pp. 653–656, doi: 10.1109/AINIT61980.2024.10581699.
[13] J. Zhao, H. Li, and F. Liu, "Neural network-based traffic matrix prediction incorporating inter-flow correlations for optical network-on-chip (ONoC)," in *2023 International Joint Conference on Neural Networks*, IEEE, Jun. 2023, pp. 1–8, doi: 10.1109/IJCNN54540.2023.10191395.
[14] H. Zheng and A. Louri, "An energy-efficient network-on-chip design using reinforcement learning," in *56th Annual Design Automation Conference 2019*, 2019, pp. 1–6, doi: 10.1145/3316781.3317768.
[15] K. Khan and S. Pasricha, "A reinforcement learning framework with region-awareness and shared path experience for efficient routing in networks-on-chip," *IEEE Design and Test*, vol. 40, no. 6, pp. 76–85, Dec. 2023, doi: 10.1109/MDAT.2023.3306719.
[16] M. F. Reza and T. T. Le, "Reinforcement learning enabled routing for high-performance networks-on-chip," in *2021 IEEE International Symposium on Circuits and Systems*, May 2021, pp. 1–5, doi: 10.1109/ISCAS51556.2021.9401790.
[17] S. Li, M.-S. Lin, W.-C. Chen, and C.-C. Tsai, "High-bandwidth chiplet interconnects for advanced packaging technologies in AI/ML applications: challenges and solutions," *IEEE Open Journal of the Solid-State Circuits Society*, vol. 4, pp. 351–364, 2024, doi: 10.1109/OJSSCS.2024.3506694.
[18] D. T. Nguyen, T. N. Nguyen, H. Kim, and H.-J. Lee, "A high-throughput and power-efficient FPGA implementation of YOLO CNN for object detection," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 27, no. 8, pp. 1861–1873, Aug. 2019, doi: 10.1109/TVLSI.2019.2905242.
[19] X.-K. Li *et al.*, "High-efficiency reinforcement learning with hybrid architecture photonic integrated circuit," *Nature Communications*, vol. 15, no. 1, Feb. 2024, doi: 10.1038/s41467-024-45305-z.
[20] O. Chatzopoulos, G. Papadimitriou, V. Karakostas, and D. Gizopoulos, "Gem5-MARVEL: microarchitecture-level resilience analysis of heterogeneous SoC architectures," in *2024 IEEE International Symposium on High-Performance Computer Architecture*, Mar. 2024, pp. 543–559, doi: 10.1109/HPCA57654.2024.00047.
[21] N. Binkert *et al.*, "The gem5 simulator," *ACM SIGARCH Computer Architecture News*, vol. 39, no. 2, pp. 1–7, May 2011, doi: 10.1145/2024716.2024718.
[22] J. Power, J. Hestness, M. S. Orr, M. D. Hill, and D. A. Wood, "gem5-gpu: a heterogeneous CPU-GPU simulator," *IEEE Computer Architecture Letters*, vol. 14, no. 1, pp. 34–36, Jan. 2015, doi: 10.1109/LCA.2014.2299539.
[23] Y. H. Yassin, M. Jahre, P. G. Kjeldsberg, S. Aunet, and F. Catthoor, "Fast and accurate edge computing energy modeling and DVFS implementation in GEM5 using system call emulation mode," *Journal of Signal Processing Systems*, vol. 93, no. 1, pp. 33–48, Jan. 2021, doi: 10.1007/s11265-020-01544-z.
[24] L. Bossuet and C. A. L.-Nino, "Power and frequency intrinsic channels on gem5," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 72, no. 2, pp. 671–684, Feb. 2025, doi: 10.1109/TCSI.2024.3435841.
[25] M. Baharloo, M. Abdollahi, and A. Baniasadi, "System-level reliability assessment of optical network on chip," *Microprocessors and Microsystems*, vol. 99, Jun. 2023, doi: 10.1016/j.micpro.2023.104843.

BIOGRAPHIES OF AUTHORS



Dr. Smriti Srivastava holds a Ph.D. in Computer Science and Engineering and has over 15 years of teaching experience. She is currently working as an associate professor at RV College of Engineering. Her areas of interest include wireless networks, NoC, and ML. She can be contacted at email: smritis@rvce.edu.in.



Dr. Minal Moharir is IEEE and ACM member, head of department at cyber security program in Department of Computer Science and Engineering, RV College of Engineering, Bengaluru, India. She have teaching experience of 23 years. Her specialization includes computer networks and security, machine/deep learning, image processing, and parallel computing. She has published more than 50 papers in reputed journals/conferences. She has also executed sponsored projects worth INR 50 lakhs funded from various agencies nationally and internationally. She can be contacted at email: minalmoharir@rvce.edu.in.



Prabu Jayant is an undergraduate student in Computer Science with a focus on ML and network security at Department of Computer Science and Engineering, RV College of Engineering, India. His research spans intrusion detection systems, network traffic classification, and the application of ML in cybersecurity frameworks. His core research interests include ML algorithms for threat prediction, statistical modeling for network anomaly detection, data-driven security solutions, and the integration of AI in intelligent defense mechanisms. He can be contacted at email: prabujayant.cy22@rvce.edu.in.



Bandaru Jnyanadeep is an undergraduate student in Computer Science with a focus on cyber-security and ML at Department of Computer Science and Engineering, RV College of Engineering, India. His research spans malware analysis, intrusion detection systems, and the application of AI in security frameworks. His core research interests include privacy-preserving data publishing, ML applications in cybersecurity, and the use of quantum cryptography to secure next generation information systems. He can be contacted at email: bjnyanadeep.cy22@rvce.edu.in.



Bipin Raj Channabasappa is pursuing a bachelor's degree in Computer Science, with a focus on cybersecurity and data privacy at Department of Computer Science and Engineering, RV College of Engineering, India. As head of his college's cybersecurity club, he leads projects on malware analysis, computer networks, neural networks, and ML. His research interests include privacy preserving data publishing, machine learning, and quantum cryptography applications in cybersecurity. He can be contacted at email: bipinrajc.cy22@rvce.edu.in.