

Using Differential Evolution for Minimal Routing in Network-on-Chips with 2D and 3D Mesh Topology

Maamar Bougherara[1,2(✉)], Nadia Nedjah[3], Djamel Bennouar[1], and Luiza de Macedo Mourelle[4]

[1] LIM Laboratory, Bouira University, Bouira, Algeria
bougherara.maamar@gmail.com
[2] Département d'Informatique, École Normale Supérieure Kouba, Algiers, Algeria
[3] Department of Electronics Engineering and Telecommunications, State University of Rio de Janeiro, Rio de Janeiro, Brazil
[4] Department of Systems Engineering and Computation, State University of Rio de Janeiro, Rio de Janeiro, Brazil

Abstract. Network-on-Chip (NoC) represents an innovative approach to interconnecting components in System-on-Chip (SoC) designs, enabling the integration of complex elements. However, being a relatively new technology, it requires substantial research efforts, particularly to streamline and accelerate the design process. NoCs can be organized in various topologies, including ring, mesh, and torus. Since multiple data packets can be transmitted simultaneously, an effective routing strategy is essential to prevent congestion and minimize delays. This paper investigates the performance of a Differential Evolution-based routing algorithm in 2D and 3D mesh NoC topologies. The proposed routing strategy aims to minimize total packet transmission latency between tasks. Simulation results demonstrate the effectiveness of this algorithm, highlighting its performance compared to other routing techniques.

Keywords: Network on Chip · Routing · Differential Evolution · Congestion

1 Introduction

Network-on-Chip (NoC) has emerged as a promising solution to address communication bottlenecks within chips, functioning like a general network but constrained by limited resources. A NoC is specifically tailored to support a particular application. These types of applications are often constrained by the number of tasks that are integrated into Intellectual Property (IP) blocks.

The process of designing a NoC-based system involves several key stages, including task assignment to IP blocks [1], the allocation of IP cores [2], and the implementation of static routing [3]. Task assignment is the phase in which each task (or group of tasks) is allocated to a suitable IP block, chosen from a pre-existing repository. Mapping, on the other hand, refers to the process of placing the assigned IP blocks onto the nodes of

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2026
O. Gervasi et al. (Eds.): ICCSA 2025 Workshops, LNCS 15888, pp. 3–18, 2026.
https://doi.org/10.1007/978-3-031-97596-7_1

the NoC's communication infrastructure. Finally, routing determines the switches to be used for establishing communication paths between the cores.

Each of these stages is optimized using computer-aided tools, commonly referred to as Electronic Design Automation (EDA). An ideal EDA tool takes the application specification as input and produces a complete system design. In practical scenarios, the EDA tool refines both hardware and software characteristics to arrive at a solution that satisfies the design requirements. This optimization process is carried out in an iterative manner. The main phases of NoC design are illustrated in Fig. 1.

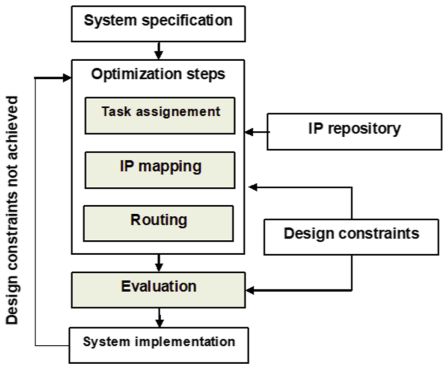


Fig. 1. Typical embedded system design flow for NoC platform.

In NoC architectures, communication delays can arise during congestion, when multiple packets attempt to pass through the same switch simultaneously. To mitigate this issue and enhance packet delivery speed, thereby improving overall system performance, this paper delves into route optimization. Specifically, it examines the static routing phase in NoC design, employing a genetic algorithm approach. The focus of this study is on routing within NoCs utilizing a 2D and 3D mesh topology.

The structure of the paper is as follows: Sect. 2 discusses 3D NoC topology and routing. Section 3 reviews prior research on routing algorithms. The routing approach is outlined in Sect. 4. In Sect. 5, we analyze the results of the simulation experiments. Finally, Sect. 6 presents the conclusion of the study and suggestions for future work.

2 Network Topology and Routing Problem

A Network-on-Chip (NoC) consists of four main components: resources (IPs), network interfaces (NIs), routers, and physical links. IPs include various components such as processors, memory, and controllers, which connect to routers via NIs.

The NI serves as a bridge between the NoC and IP protocols, separating computation from communication to enhance flexibility and IP reuse. Routers manage data transmission and arbitration between multiple port requests, while physical links establish logical connections between IPs, NIs, and routers. A tile in the NoC consists of an IP, an NI, and a router.

One of the simplest ways to connect tiles is through network topologies. Among them, the 2D mesh topology is the most widely adopted in NoCs due to its simplicity and scalability. Figure 2 illustrates the architecture of 2D Mesh Network on Chip.

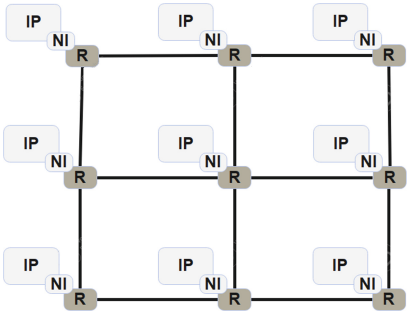


Fig. 2. 2D Mesh-based NoC.

However, two-dimensional NoCs often fail to meet the diverse requirements of System-on-Chip (SoC) designs. To address this, three-dimensional Network-on-Chip (3D NoC) architecture provides an effective solution to the interconnection complexities inherent in large-scale SoCs. By leveraging integrated circuit stacking technology, 3D NoC overcomes the performance and scalability limitations faced by traditional two-dimensional NoCs.

A three-dimensional (3D) mesh network is created by stacking multiple layers of 2D meshes and establishing vertical links for interlayer communication. One promising solution for these vertical interconnections is Through-Silicon Via (TSV) technology [5]. Each router in the network is responsible for buffering and routing messages between connected tiles. These routers can connect to up to six neighboring routers. This design allows one channel to transmit data while another channel buffers incoming data, enhancing overall efficiency. Figure 3 provides an illustration of the 3D mesh-based Network-on-Chip (NoC) architecture.

Routing algorithms play a vital role in determining the path a packet will follow from its source to its destination, and selecting the right algorithm is crucial when designing a 3D-NoC. Various challenges arise in NoC routing, such as congestion, deadlock, livelock, throughput, power consumption, delay, and more. The routing algorithm aims to optimize network performance by minimizing congestion and reducing communication delays between channels. Additionally, it considers factors like buffer size to select the best path, ensuring smooth traffic flow and preventing bottlenecks. Given the significance of these factors, routing algorithms have been the focus of extensive research since the introduction of NoCs. Below, we review some of the key studies on 3D routing.

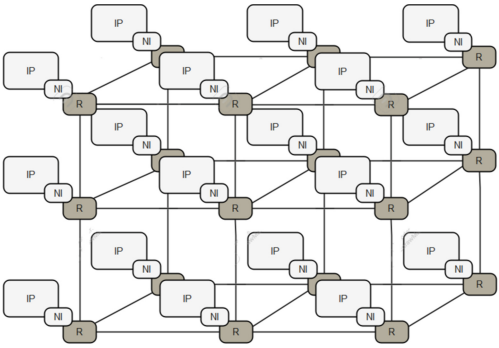


Fig. 3. 3D Mesh-based NoC with 27 resources.

3 Related Work

While numerous routing algorithms have been proposed for NoC systems, most focus primarily on 2D-NoC topologies. In the case of 3D-NoCs, relatively few studies have specifically addressed routing algorithms, which can be categorized into two main groups. The first group consists of well-established 2D-NoC routing schemes that have been adapted to the third dimension, such as Dimension Ordered Routing (DOR) [6], Valiant [7], O1TURN [8], and Odd-Even [9]. The second group is different from 2D NoCs, as only a few congestion-aware routing algorithms have been proposed for 3D NoCs, making it a less explored research area. DyXYZ [10] is an adaptive routing algorithm designed to address congestion issues in 3D NoCs. It uses congestion information from neighboring nodes to make handover decisions at the current node. The 3D-FAR algorithm, introduced in [11], is a regionally congestion-sensitive routing approach that organizes a clustering structure to gather and propagate congestion data. MAR, presented in [12], is an adaptive routing scheme that routes packets based on Hamiltonian paths, avoiding the need for virtual channels. In [13], the Global A (G-CARA) congestion-aware routing algorithm is introduced for traffic management in 3D NoCs. It effectively routes data packets to regions with lower traffic, preventing network congestion. FT-DyXYZ, discussed in [14], is an adaptive, fault-tolerant routing algorithm designed to handle permanent faulty links in 3D mesh NoCs. It uses proximity-based congestion information to balance the traffic load.

Metaheuristics have been employed as intelligent methods for creating efficient routing algorithms. One such approach is AntNet [15], a dynamic routing strategy for telecommunication networks. The routing problem is addressed in [16] for three different 3D NoC topology (mesh, torus and hypercube), based on the ant conlony metaheuristic. In [17,18], a routing algorithm aimed at minimizing congestion is presented, utilizing Genetic Algorithm and Particle Swarm Optimization (PSO). The results show that PSO can achieve better performance, especially when the paths are deterministic, in both 2D and 3D NoCs.

4 Routing Algorithm Approach

The encoding of the routing problem relies on the definition and use of certain essential objects. Before presenting the routing algorithm, these objects must be defined, along with the key parameters, notably the concept of a solution:

1. Path: A path between a source (Src) and a destination (Dst) is a set of intermediate nodes that ensures the transport of packets from the source to the destination. There are two types of paths: minimal and non-minimal.
   - A path is said to be minimal if it connects a source with coordinates (x_i, y_i, z_i) to a destination with coordinates (x_j, y_j, z_j) with a length equal to the Manhattan distance.
   - A path is said to be non-minimal if its length is determined by Eq. 1:

$$\text{length}(r_i, r_j) = \text{nLinks}(r_i, r_j) + x, \tag{1}$$

   where x is a user-defined parameter that limits the number of additional hops required to construct a non-minimal path.
2. Set of paths: For optimal routing, the pair (Src, Dst) represents a communication between a source Src and a destination Dst, and it should not be based on a single path. It is necessary to create several different paths of the same pair, in order to be able to choose the efficient path. Despite the random aspect of our approach, we proposed to generate s different and random paths between each pair (Src, Dst). Among which, the best path will be kept in the final set of paths of the solution. The Table 1 shows 5 possible paths for the same pair (Src, Dst).

Table 1. Multiple paths of the same pair

Pair (Src, Dst)	Path ID	Path
(13, 8)	1	[13, 9, 5, 6, 7, 8]
	2	[13, 9, 10, 11, 12, 8]
	3	[13, 14, 15, 16, 12, 8]
	4	[13, 14, 10, 11, 7, 8]
	5	[13, 9, 10, 6, 7, 8]

3. Complete solution: A complete solution includes the set of paths connecting several pairs of nodes concerned by routing according to the type of application to implement in the NoC. In the first step, the choice of the best path among the s generated paths is done randomly. The algorithm then takes over to select the optimal solution based on the objective. Table 2 shows an example of a complete solution chosen among the paths that exist on each of the (Src, Dst) pairs.

The final solution obtained at this level consists of a set of paths connecting the different pairs, (Src, Dst), according to the type of application. These paths are saved in a file in the form of routing tables, used by the router.

Table 2. Complete routing solution

Pair (Src, Dst)	Chosen Path ID	Path Detail
(6, 16)	4	[6, 10, 11, 12, 16]
(1, 11)	1	[1, 2, 3, 7, 11]
(10, 4)	5	[10, 11, 12, 8, 4]
(9, 15)	2	[9, 10, 14, 15]
(13, 8)	3	[13, 14, 15, 16, 12, 8]

Routing optimization in NoCs aims to reduce communication latency, thereby shortening the execution time of embedded applications. For a set of simultaneous communications, two main objectives must be minimized: packet transmission delay and the number of nodes traversed between the source and destination.

Congestion occurs when incoming traffic exceeds the network's capacity, leading to queuing delays, packet loss, and communication blockages [19]. Increasing buffer size can help mitigate congestion but introduces area overhead, while limiting buffer size degrades Network-on-Chip (NoC) performance.

A congestion scenario arises when a single communication channel must handle multiple packets from different sources simultaneously, causing delays despite the use of minimal paths. To address this issue, adaptive routing algorithms consider both node positions and real-time network load, selecting less congested paths even if they are non-minimal.

Congestion is quantified as a cost reflecting network link usage. Initially set to zero, the cost increases as links are used. The congestion cost function for a path of length L is expressed as follows:

$$Congestion_path_i = \sum_{j=1}^{links(i)} C_{i,j}, \tag{2}$$

where $C_{i,j}$ represents the congestion cost of the j^{th} link. The total network congestion is:

$$Congestion(S) = \sum_{i=1}^{l} Congestion_path_i. \tag{3}$$

4.1 Differential Evolution

Differential Evolution (DE), proposed by Price and Storn in 1997 [20], is an evolutionary algorithm related to genetic algorithms but uses different operators and mechanisms. It is an algorithm based on an initial population like genetic algorithms. In differential evolution, not only the operations are used with a different order, but the content of the operations also changes. The main distinction between DE and genetic algorithms resides in their exploration mechanisms. While genetic algorithms depend mainly on crossover to produce new solutions, DE is driven by mutation, which is based on the

difference vectors computed from randomly selected solution pairs. The search process of the algorithm is driven by the mutation operator, whereas the selection mechanism directs the convergence towards optimal regions within the search space. The algorithm notation ($DE/x/y/z$) is generally used to define a DE strategy [21] such that: The variable x identifies the target vector for mutation, which may correspond to a randomly selected population member or the best individual of the current generation. The parameter y specifies the number of differential vectors used to perturb the target vector, and z denotes the adopted crossover scheme, typically chosen between exponential and binomial variants.

4.2 DE Algorithm Steps

Differential Evolution (DE) is a population-based global optimization algorithm. It initializes a population composed of N individuals, each represented as a D-dimensional vector. Every individual corresponds to a potential solution, $X_{i,t} = \{x^1_{i,t}, .., x^D_{i,t}\}$, $i = 1, .., N$, where G indicates the generation to which the population belongs [22]. The initial population is randomly generated from the entire search space. The main steps of the DE algorithm are as follows [23]:

1. Mutation operation: The mutation operator is responsible for generating diversity within the population by producing a mutant vector $v_{i,t}$ for each individual $x_{i,t}$ at generation t. This operator is designed according to specific strategies, among which the most widely used are reported in Table 3 [24].

Table 3. Mutation strategies used

Mutation	Function
DE/rand/1	$v_{i,t} = x_{r_1,t} + F.(x_{r_2,t} - x_{r_3,t})$
DE/best/1	$v_{i,t} = x_{best,t} + F.(x_{r_1,t} - x_{r_2,t})$
DE/best/2	$v_{i,t} = x_{best,t} + F.(x_{r_1,t} - x_{r_2,t}) + F.(x_{r_3,t} - x_{r_4,t})$
DE/rand/2	$v_{i,t} = x_{r_1,t} + F.(x_{r_2,t} - x_{r_3,t}) + F.(x_{r_4,t} - x_{r_5,t})$
DE/current-to-best/2	$v_{i,t} = x_{i,t} + F.(x_{best,t} - x_{r_1,t}) + F.(x_{r_2,t} - x_{r_3,t})$
DE/current-to-rand/2	$v_{i,t} = x_{i,t} + F.(x_{r_1,t} - x_{r_2,t}) + F.(x_{r_3,t} - x_{r_4,t})$

   The mutant vector $v_{i,t}$ is obtained through the mutation process. The integers r_1, r_2, r_3, r_4 and r_5 are randomly from $[1..N]$ and are distinct from the current index j. The vector $x_{best,t}$ represents the best solution identified at generation t. The scaling factor FF is a real constant, generally selected within the interval $[0..1]$, which adjusts the magnitude of the differential variation.

2. Crossover operation: This operation enhances the population diversity following the mutation phase. Subsequently, the crossover operator combines components of the mutant vector vi,tvi,t and the target vector xi,txi,t to generate the trial vector ui,tui,t. The crossover mechanism is mathematically defined in Eq. 4 [25]:

$$u_{i,t}^{j} = \begin{cases} v_{i,t}^{j} \text{ if} & (rand_j \leq CR) or (j = j_{rand}) \\ x_{i,t}^{j} \text{ otherwise,} \end{cases} \tag{4}$$

where $j = 1, 2, .., D$ and $rand_j$ denotes the j th evaluation from a uniform random number generator over the interval $[0..1]$ [26]. The crossover rate CR is a user-defined parameter within $[0..1]$, and j_{rand} represents a randomly chosen integer in the range $[1..D]$ [27].

3. Selection operation:

In order to preserve a constant population size throughout the evolutionary process, the selection phase is applied. Specifically, the trial vector is first evaluated according to the objective function and subsequently compared to its associated target vector from the current generation. In case the trial vector outperforms the target vector, it substitutes the target vector within the population, otherwise, the target vector is preserved. The selection operation is represented in Eq. 5 [28]:

$$x_{i,t+1} = \begin{cases} u_{i,t} \text{ if} & f(u_{i,t}) \leq f(x_{i,t}) \\ x_{i,t} \text{ otherwise,} \end{cases} \tag{5}$$

Each generation goes through the three steps of mutation, crossover, and selection until a stopping criterion is reached. The DE algorithm, which groups the steps presented above, is illustrated in Algorithm 1. The parameters used in the algorithm include the number of individuals (N), the maximum number of iterations (Max), the choice of mutation scheme, and the initialization of the scaling factor (F) and crossover rate (CR). The F parameter controls the mutation intensity and is applied during the mutation phase, while CR defines the probability of gene exchange and is used in the crossover phase. These parameters play a crucial role in balancing exploration and exploitation to enhance the algorithm's performance.

In [25,28], the "rand" strategy was used for both assignment and mapping problems. Parameters such as the mutation factor (F), crossover rate (CR), and number of iterations were carefully calibrated, leading to better results compared to other strategies. These optimized parameters were adopted for the routing optimization phase to ensure consistent performance. We run the algorithm for 200 iterations with a swarm of 100 individuals. The cognitive scale factor is set to 0.5, and the crossover coefficient is 0.9. The mutation strategy used is based on the rand/1.

Algorithm 1. The main steps of DE

```
generate the individuals of the population
evaluate the individuals with the objective function
initialize the best global solution g
t ← 0
while t < max of iteration do
   for each individual i do
      generate the mutant vector using the mutation operations described in Table 3
      generate the test vector using the crossover operation described by Equation 4
      select the individual using the selection operation described by Equation 5
      if individual is better than g then
         g ← the individual
      end if
   end for
   t ← t + 1
end while
return best solution g
```

5 Simulations and Performance Results

In this section, we compare the results obtained using two deterministic routing algorithms: XY [31] and Odd-Even [9] for the 2D topology, and XYZ [32] and Odd-Even [9] for the 3D topology. The objective of this comparison is to evaluate the performance of these algorithms against a metaheuristic-based algorithm using Particle Swarm Optimization (PSO) [18], as well as our algorithms based on Differential Evolution (DE). We used the Access-Noxim simulator [29], an enhanced version of Noxim. Noxim is a Network-on-Chip (NoC) simulation tool [30], written in C++ and based on the SystemC library. Its flexibility allows for easy modification of the routing strategy by writing and testing code in C++.

In this study, we focus on latency, which is defined as the time elapsed between the injection of a packet at the local port and the consumption of the last data at the local port of the destination node.

We studied networks with the following topologies: 2D and 3D mesh with a size of 5×5 (a 2D mesh with 25 nodes) and $3\times3\times3$ (a 3D mesh with 27 nodes). Four routing algorithms were tested in the simulator: DE, the algorithm we evaluated, along with DOR (XYZ), Odd-Even, and PSO. The parameters related to the complete system simulation configuration are detailed in Table 4.

The tests are performed by adjusting the Traffic Models and Packet Injection Rates (PIR). A source-destination pair is defined as the node that sends a packet and the node that receives it. These pairs are chosen randomly based on various distribution models, which fall into two primary categories: deterministic models and random models.

Under deterministic models, each node is assigned a specific destination based on its position. Examples of these models include Complement, Transpose Matrix 1, and

Table 4. Parameter setting

Parameter	Setting
Network	2D Mesh vs 3D Mesh
Network dimensions	5x5 vs 3x3x3
Packet size	8
Virtual channels	4
Warm-up time	1000
Simulation time	20,000 cycles
Routing algorithms	DE, PSO, XYZ and ODD-Even

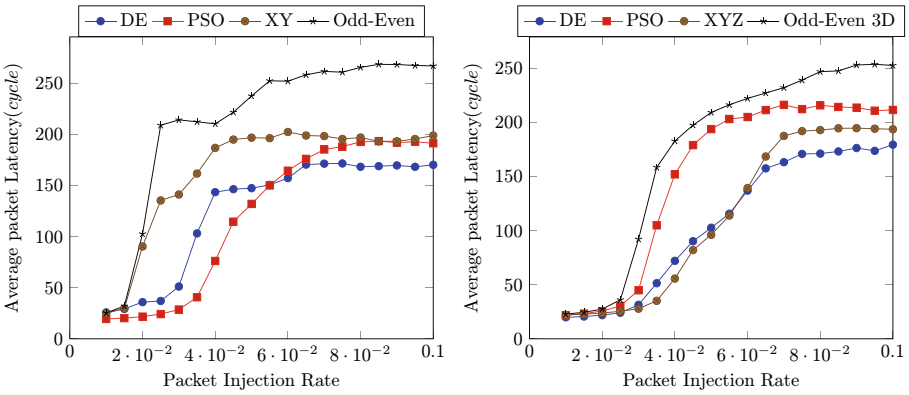
Table 5. Destination nodes of deterministic patterns.

Network	Pattern	source node	Destination node
2D Mesh	Complement	(x,y)	(size - x - 1,size - y - 1)
	Transpose 1	(x,y)	(size - y - 1,size - x - 1)
	Transpose 2	(x,y)	(y,x)
3D Mesh	Complement	(x,y,z)	(size - x - 1, size - y - 1, size - z - 1)
	Transpose 1	(x,y,z)	(size - y - 1, size - z - 1, size - x - 1)
	Transpose 2	(x,y,z)	(y,z,x)

Transpose Matrix 2 [16]. The relationship between source nodes and their corresponding destinations is detailed in Table 5.

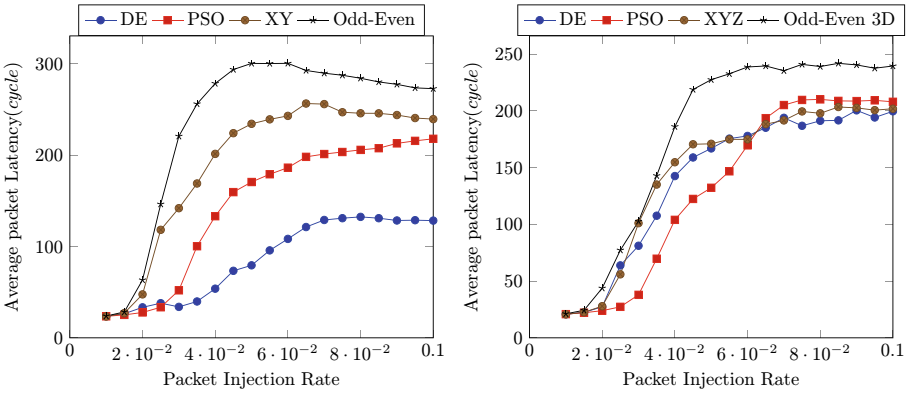
In random models, destinations are selected based on probability. In the Rand model, every node has an equal likelihood of being a source or a destination. The Hotspot model ensures that all nodes have the same probability of being sources, but certain nodes, known as Hot-spots, are more likely to be chosen as destinations [16]. In the Local model, any node can act as a source, but destinations are restricted to its immediate neighboring nodes, which are selected randomly.

For the routing schemes, the results of this analysis represent the latency obtained for different packet injection rates. These results allow us to observe how latency varies as the injection rate increases or decreases, providing a comprehensive overview of routing performance. Figure 4 corresponds to the Rand routing scheme, where packets are generated randomly across the network. Here, we present the results achieved by the four algorithms for both 2D and 3D routing. In the analysis of 2D routing, the PSO and DE algorithms achieved better results compared to the XY and Odd-Even algorithms. However, PSO performed better than DE. Nevertheless, when the packet injection rate (PIR) reached 0.06, DE outperformed PSO. On the other hand, in the case of 3D routing, the results highlight that the DE algorithm delivers better performance in terms of latency compared to PSO, XYZ, and Odd-Even for random patterns. It is also noted that PSO does not achieve better results than DE, while DE is superior to XYZ, and XYZ is superior to PSO. Figure 5 illustrates the latency results for both 2D and 3D



(a) 2D (b) 3D

Fig. 4. Latency results obtained for the Rand routing scheme.

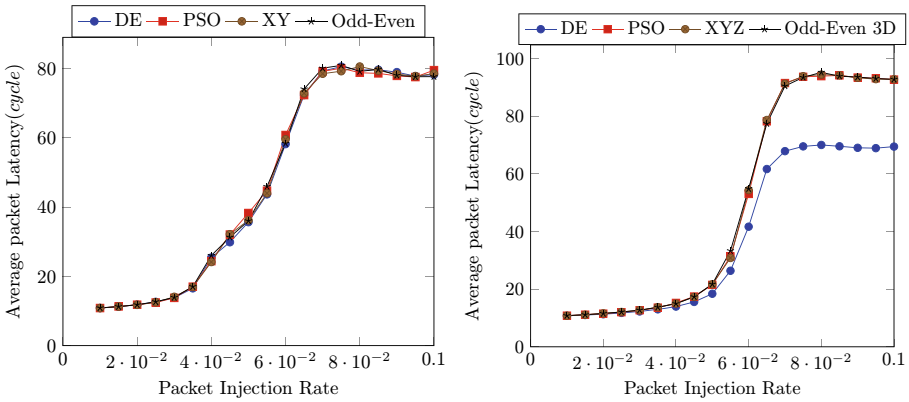


(a) 2D (b) 3D

Fig. 5. Latency results obtained for the Hotspot routing scheme.

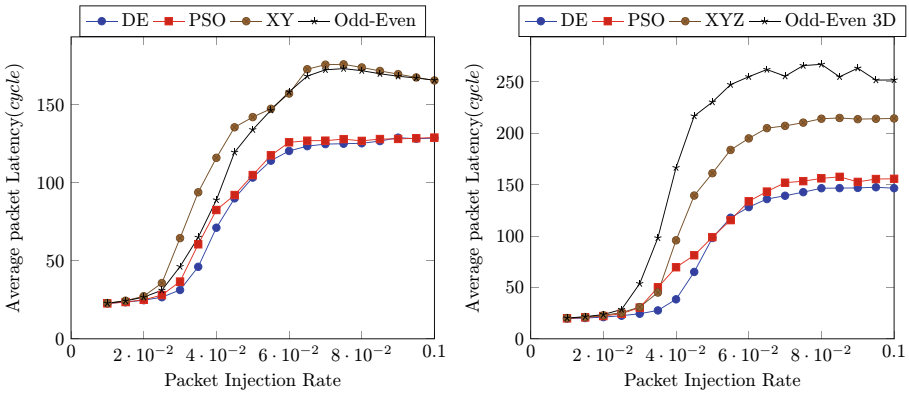
in the Hotspot routing scheme, where certain network nodes generate higher traffic than others.

In the case of the Hotspot traffic in a 2D NoC, the PSO and DE algorithms outperform the XY and Odd-Even algorithms, demonstrating their efficiency in handling high-intensity traffic. For the 3D NoC, PSO initially delivers strong latency performance until the PIR reaches 0.065. Beyond this point, DE surpasses both PSO and XYZ, while PSO continues to perform better than XYZ. Finally, Fig. 6 presents the latency curves for the Local routing scheme, where packets are generated locally by each node for both 2D and 3D routing. In 2D routing, all routing algorithms produce similar latency curves, indicating that their performance is comparable for this specific scenario, with no significant differences in latency. For 3D routing, XYZ, Odd-Even, and PSO exhibit almost identical latency curves, whereas DE outperforms the other



(a) 2D (b) 3D

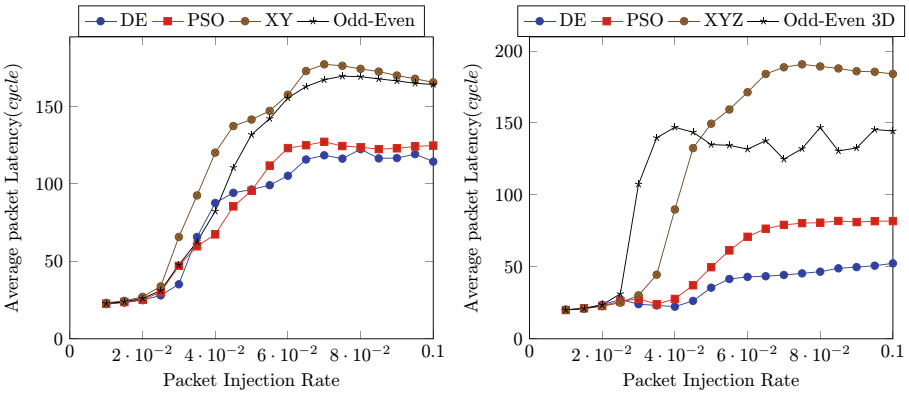
Fig. 6. Latency results obtained for the Local routing scheme.



(a) 2D (b) 3D

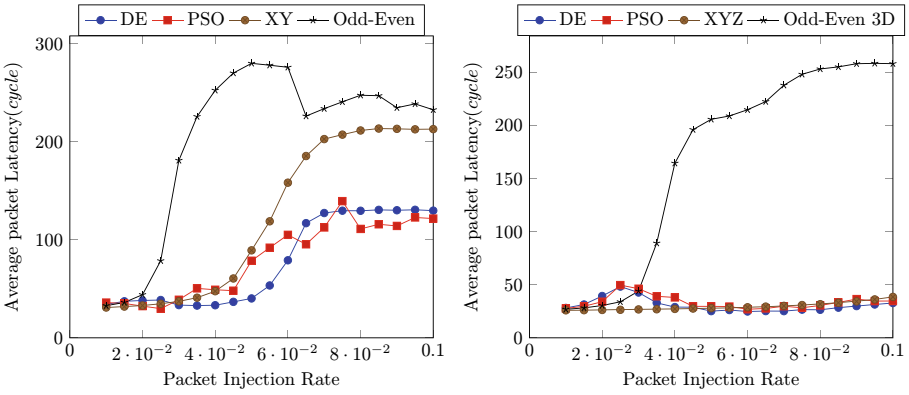
Fig. 7. Latency results obtained for the Transpose_1 routing scheme.

three algorithms. In deterministic routing schemes, Fig. 7 visually presents the latency curves for the Transpose_1 routing scheme in both 2D and 3D. In the analysis of 2D routing, we observe that the PSO and DE algorithms yield similar results and outperform the XY and Odd-Even algorithms. Regarding the 3D routing results, the latency curves of PSO and DE are nearly identical and surpass those of XYZ and Odd-Even. This indicates that PSO and DE achieve better performance in this scenario compared to XYZ and Odd-Even. Figure 8 visually represents the latency curves for the Transpose_2 routing scheme in both 2D and 3D. For the Transpose_2 scheme in a 2D NoC, DE demonstrates improved latency performance when the PIR exceeds 0.045. This suggests that DE is better suited for handling higher packet injection rates in this specific pattern. In contrast, in a 3D NoC, starting from a PIR of 0.04, both PSO and DE begin



(a) 2D (b) 3D

Fig. 8. Latency results obtained for the Transpose_2 routing scheme.



(a) 2D (b) 3D

Fig. 9. Latency results obtained for the Complement routing scheme.

to show better latency results, with DE outperforming PSO. This indicates that DE is more effective than PSO in reducing latency for this particular traffic pattern. Finally, Fig. 9 visually depicts the latency curves for the Complement routing scheme in both 2D and 3D. In the Complement scheme for 2D NoC, PSO outperforms DE when the packet injection rate (PIR) exceeds 0.065, indicating that PSO adapts better to higher PIR values. For 3D routing, XYZ exhibits a flat latency curve, suggesting it does not significantly improve latency. Similarly, PSO follows a pattern close to XYZ, whereas DE delivers better results starting from a PIR of 0.045. This indicates that DE is more effective than both PSO and XYZ in reducing latency in the Complement scheme for 3D NoC.

6 Conclusion

This work introduces a routing algorithm based on Differential Evolution for Network-on-Chip (NoC) architectures in both 2D and 3D mesh topologies. Extensive testing was conducted using both deterministic and random traffic patterns, with comparisons against PSO, Odd-Even, and XYZ routing algorithms. The proposed approach consistently outperforms existing methods, achieving minimal latency compared to XY and Odd-Even in 2D, as well as XYZ and Odd-Even_3D in 3D. Despite these promising results, further improvements are necessary. Future work will explore additional metaheuristic approaches and integrate real application task graphs while evaluating NoCs with larger dimensions.

References

1. Bougherara, M., Nedjah, N., Mourelle, L., Rahmoun, R., Sadok, A., Bennouar, D.: IP assignment for efficient NoC-based system design using multi-objective particle swarm optimisation. Inter. J. Bio-Inspired Comput. **12**(4), 203–213 (2018)
2. Bougherara, M., Nedjah, N., Bennouar, D., et al.: Core/task associations for efficient application implementation on Network-on-Chip. In: 2018 International Conference on Computer and Applications (ICCA), pp. 18-22. IEEE (2018)
3. Bougherara, M., Nedjah, N., Bennouar, D., Kemcha, R., de Macedo Mourelle, L.: Efficient application mapping onto three-dimensional network-on-chips using multi-objective particle swarm optimization. In: Misra, S., et al. (eds.) ICCSA 2019. LNCS, vol. 11620, pp. 654–670. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-24296-1_53
4. Hu, J., Marculescu, R.: Energy-aware mapping for tile-based noc architectures under performance constraints. In: Proceedings of the 2003 Asia and South Pacific Design Automation Conference, pp. 233-239. ACM (2003)
5. Davis, W.R., et al.: Demystifying 3d ics: the pros and cons of going vertical. IEEE Design Test Comput. **22**(6), 498–510 (2005)
6. Sullivan, H., Bashkow, T.: A large scale, homogeneous, fully distributed parallel machine. ACM SIGARCH Comput. Architecture News **5**(7), 105–117 (1977)
7. Valiant, L., Brebner, G.: Universal schemes for parallel communication. In: Proceedings of the Thirteenth Annual ACM symposium on Theory of Computing, pp. 263–277. ACM (1981)
8. Seo, D., Ali, A., Lim, W., Rafique, N., Thottethodi, M.: Near-optimal worst-case throughput routing for two-dimensional mesh networks. In: ACM SIGARCH Computer Architecture News, vol. 33(2), pp. 432–443. IEEE Computer Society (2005)
9. Chiu, G.-M.: The odd-even turn model for adaptive routing. IEEE Trans. Parallel Distrib. Syst. **11**(7), 729–738 (2000)
10. Ebrahimi, M., Chang, X., Daneshtalab, M,, et al.: DyXYZ: fully adaptive routing algorithm for 3D NoCs. In: 2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, pp. 499-503. IEEE (2013)
11. Ebrahimi, M.: Fully adaptive routing algorithms and region based approaches for two-dimensional and three-dimensional networkson-chip. lET Comput. Digital Tech. **7**(6), 264–273 (2013)
12. Ebrahimi, M., Daneshtalab, M., Liljeberg, P., Plosila, J., Flich, J., Tenhunen, H.: "Path-based partitioning methods for 3D networks-on-chip with minimal adaptive routing. IEEE Trans. Comput. **63**(3), 718–733 (2014)

13. Nosrati, N., Shahhoseini, H.S.: G-cara: a global congestion-aware routing algorithm for traffic management in 3d networks-on-chip. In : 2017 Iranian Conference on Electrical Engineering (ICEE), pp. 2188-2193. IEEE (2017)
14. Jouybari, H.N., Mohammadi, K.: A low overhead, fault tolerant and congestion aware routing algorithm for 3D mesh-based Network-on-Chips. Microprocessors Microsyst. **38**(8), 991–999 (2014)
15. Silva, Jr., L., Nedjah, N., , De Macedo Mourelle, L.: ACO approach in static routing for Network-on-Chips with 3D mesh topology. In: 2013 IEEE 4th Latin American Symposium on Circuits and Systems (LASCAS), pp. 1-4. IEEE (2013)
16. Silva, Jr., L., Nedjah, N., De Macedo Mourelle, L.: Efficient routing in Network-on-Chip for 3D topologies. Inter. J. Electr. **102**(10), 1695–1712 (2015)
17. Bougherara, M., Amara, R.: Routing using genetic algorithm in network on chips with 3D mesh topology. In: 2023 International Conference on Computer and Applications (ICCA), pp. 1-5. IEEE (2023)
18. Bougherara, M., Nedjah, N., Bennouar, D., Mourelle, L.d.M.: Routing in 3D NoCs Using Genetic Algorithm and Particle Swarm Optimization. In: Gervasi, O., et al. (ed.) Computational Science and Its Applications – ICCSA 2023 Workshops. ICCSA 2023. LNCS, vol 14104. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-37105-9_40
19. Alfaraj, N., Zhang, J., Xu,Y., Chao, H.J.: Hope: hotspot congestion control for clos network on chip. In: Proceedings of the Fifth ACM/IEEE International Symposium on Networks-on-Chip, pp. 17–24 (2011)
20. Fleetwood, K.: An introduction to differential evolution. In: Proceedings of Mathematics and Statistics of Complex Systems (MASCOS) One Day Symposium, 26th November, Brisbane, Australia, pp. 785–791 (2004)
21. Feoktistov, V.: Differential evolution. Springer (2006)
22. Robič, T., Filipič, B.: DEMO: differential evolution for multiobjective optimization. In: Coello Coello, C.A., Hernández Aguirre, A., Zitzler, E. (eds.) EMO 2005. LNCS, vol. 3410, pp. 520–533. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-31880-4_36
23. Bougherara, M., Amara, R., Kemcha R.: DEMAP: differential evolution mapping for network on chip optimization. Inter. J. Robot. Autom. (IJRA) **12**(4), 394-404
24. Das, S., Suganthan, P.N.: Differential evolution: a survey of the state-ofthe-art. IEEE Trans. Evolutionary Comput. **15**(1), 4–31 (2010)
25. Bougherara, M., Nedjah, N., Bennouar, D., Kemcha, R., de Macedo Mourelle, L.: Application mapping onto 3D NoCs using differential evolution. In: Gervasi, O., et al. (eds.) ICCSA 2020. LNCS, vol. 12251, pp. 89–102. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58808-3_8
26. Madavan, N.K.: Multiobjective optimization using a pareto differential evolution approach. In: Proceedings of the 2002 Congress on Evolutionary Computation. CEC 2002 (Cat. No. 02TH8600), vol. 2, p. 1145–1150. IEEE (2002)
27. Ahmad, M.F., Isa, N., Lim, W.H., et al.: Differential evolution: a recent review based on state-of-the-art works. Alexandria Eng. J. **61**(5), 3831–3872 (2022)
28. Bougherara, M., Kemcha, R., Nedjah, N., et al.: IP assignment optimization for an efficient noc-based system using multi-objective differential evolution. In: International Conference on Metaheuristics and Nature Inspired Computing (META), pp. 435-444 (2018)
29. Catania, V., et al.: Noxim: an open extensible and cycle-accurate network on chip simulator. In: IEEE 26th International Conference on Application-specific Systems Architectures and Processors (ASAP), pp. 162–163 (March 2015)
30. Access Noxim. http://access.ee.ntu.edu.tw/noxim/index.html

31. Alikhah-Asl, E., Reshadi, M.: XY-axis and distance based NoC mapping (XY-ADB). In: 2016 8th International Symposium on Telecommunications (IST), pp. 678–683. IEEE (2016)
32. Ahmed, A.B., Abdallah, A.B.: LA-XYZ: low latency, high throughput lookahead routing algorithm for 3D Network-on-Chip (3D-NoC) architecture. In: 2012 IEEE 6th International Symposium on Embedded Multicore SoCs, pp. 167–174. IEEE (2012)