

Routing in 3D NoCs Using Genetic Algorithm and Particle Swarm Optimization

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

[1] LIMPAF Laboratory, Bouira University, Bouira, Algeria
[2] Département d'Informatique, Ecole 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
LDMM@ENG.UERJ.BR

Abstract. Networks-on-chip are a new concept in System-on-chip interconnections, facilitating and optimizing complex components integration. However, as it is a new technology, it still requires some research, especially concerning the acceleration and simplification of design phases. Networks-on-chip can be arranged in different topologies, such as hypercube, mesh and torus. Since several data packets can be transmitted simultaneously through the network, an efficient routing strategy must be used in order to avoid congestion delays. In this paper, we propose and evaluate the performance of two routing methods, based on genetic algorithm and particle swarm optimization, for Networks-on-chip with 3D mesh topology. The routing is driven by the minimization of total latency in packets transmission between tasks. The simulation results show that the routing based on genetic algorithm and particle swarm optimization outperforms other routing algorithms in terms of latency.

Keywords: 3D Networks-on-chip · Routing · Genetic algorithm · Particle swarm optimization · Congestion

1 Introduction

Networks-on-Chip (NoCs) provide an infrastructure for on-chip communication, in order to target bottlenecks when running parallel applications. It is based on the well known area of computer network, but driven by the limited resources that a chip design imposes. The basic elements of a NoC are resources (R) and switches (S), connected by links. A resource can be a processor, memory or any kind of functional component [1]. A switch is a component that provides the communication between resources, through links, where packets of information flow.

The topology of the NoC defines the way links are interconnected, which can have the shape of a mesh, torus or hypercube [2]. Figure 1 shows the architecture of a 3D NoC, based on the mesh topology, where, for example, RNI stands for Resource Network Interface, D for Digital Signal Processor, M for Memory, C for Cache, P for Processor, FP for Floating-Point unit and Re for a Reconfigurable component.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023
O. Gervasi et al. (Eds.): ICCSA 2023 Workshops, LNCS 14104, pp. 601–613, 2023.
https://doi.org/10.1007/978-3-031-37105-9_40

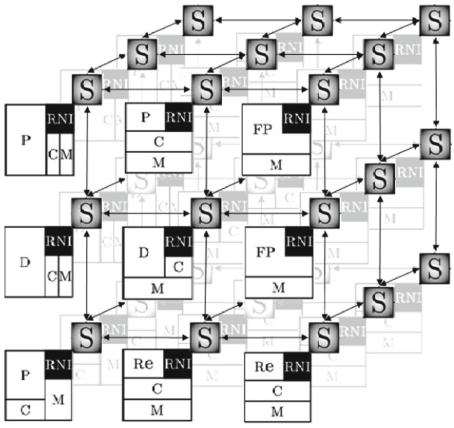


Fig. 1. 3D mesh-based NoC with 27 resources

A NoC is optimised to run a specific application, which is divided into tasks implemented by Intellectual Property (IP) blocks. The design process can be divided into intermediate steps, such as IP assignment of tasks [3] and IP mapping [4,22]. IP assignment consists of associating each task (or set of tasks) to an appropriate IP block, selected among those that are readily available in a repository. The mapping of an application consists of associating the set of assigned IPs to each node in the communication infrastructure of the NoC. Routing defines which paths will be used to establish the communication between IPs. Each of these steps is optimized by computer-aided tools, which use the application specification and generates a full system implementation. These tools optimize the characteristics of the hardware and software in order to obtain a solution that meets the design specifications, as seen in Fig. 2 for a NoC design.

In a NoC architecture, communication delays can occur in congestion situations, when multiple packets require communication through the same switch at the same time [16]. In order to avoid this congestion problem, and thus speed up the overall runtime system, our work explores routing optimization. This paper describes the routing optimization step in the design of NoCs, using static routing based on Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) for NoCs with 3D mesh topology.

The rest of the paper is organized as follows. Previous works on routing algorithms are introduced in Sect. 2. The overview of our routing approach is presented in Sect. 3. In Sect. 4, we discuss the results of the simulation experiments. The conclusion of this work and future work are presented in Sect. 5.

2 Related Work

Many routing algorithms have been proposed for NoCs, but most of them focus only on 2D NoC topologies. Among all the studies conducted for 3D NoCs, few of them

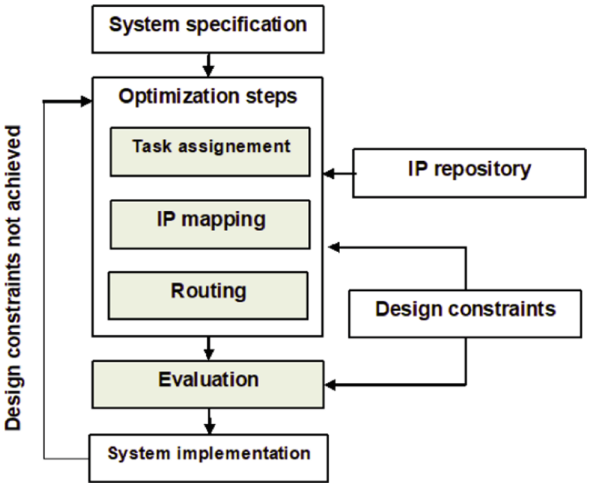


Fig. 2. Typical embedded system design flow for NoCs

targeted routing algorithms and they can be classified into two categories. The first one includes some of the well known routing schemes in 2D NoCs, that were extended to 3D, such as Dimension Ordered Routing (DOR) [5], Valiant [6], O1TURN [7] and Odd-Even [8]. The second one includes some specially related to 3D, such as DyXYZ [9], 3D-FAR [10], MAR [11], Global A (G-CARA) [12], FT-DyXYZ [13].

In 2D mesh topology, the most popular DOR routing algorithm is XY. In this algorithm, each node is characterized by a pair (x,y), representing its position in the network (row and column). By applying the XY algorithm, the packet is, first, routed along the x axis until reaching the destination X and, then, routes along the y axis, until reaching the destination Y. It is a deterministic and minimal algorithm known for its simplicity of implementation. However, it becomes ineffective as soon as the network is overloaded. In this situation, it is not able to avoid congested regions of the network. Therefore, the packets must wait for the channels to become free in order to be transmitted.

The XYZ routing algorithm is the corresponding XY routing algorithm in a 3D mesh topology. Each node is characterized by a triple (x, y, z). The packet is first routed along the X axis, then the Y axis, and finally the Z axis to reach the destination. Although the XYZ algorithm obtains the shortest path, it is not able to avoid congested regions of the network.

The routing algorithm offered by Valiant achieves optimal throughput by balancing the load globally across the entire network. Routing consists of two phases: first, packets are routed from the source to a random intermediate node, followed by routing to the destination. Both phases used DOR algorithm.

O1Turn routing achieves both minimum length and near optimal throughput by simply choosing between 2D turn paths: XY or YX. O1Turn can be extended in 3D topology. In this case, you can choose between several possibilities such as: XYZ, XZY, ZXY, ZYX.

The OE routing protocol, also known as "Odd-Even", uses partially adaptive routing without virtual channels. It relies on specific rules based on column parity to determine allowed and unallowed turns, which helps to avoid circular loops and unwanted blocking states.

Unlike 2D NoCs, a few congestion-aware routing algorithms were proposed for 3D NoCs. DyXYZ [9] applies adaptive routing to solve 3D NoC congestion problems, using congestion information from neighbouring nodes to make handover decision at the current node. 3D-FAR is a regional congestion sensitive routing algorithm proposed in [10]. It presents a clustering structure to collect and propagate congestion information. MAR [11] is an adaptive routing algorithm that routes packets based on Hamiltonian paths without using virtual channels. In [12], a Global A (G-CARA) congestion sensitive routing algorithm is proposed for traffic management in 3D NoC. The algorithm efficiently transmits data packets to regions with low traffic to avoid network congestion. FT-DyXYZ [13] is an adaptive fault tolerant routing algorithm, with the ability to tolerate permanent faulty links in 3D mesh NOCs that uses proximity congestion information in order to balance traffic.

Computational intelligence meta-heuristics have been used as intelligent approaches for efficient routing algorithms. This is the case of AntNet [14], a dynamic routing scheme for telecommunication networks. In [15], the authors applied Ant Colony Optimization (ACO) technique to address the routing problem in 3D NoCs based on three topologies: mesh, torus and hypercube. The results were better than those obtained by deterministic and minimal routing algorithms, such as DOR.

3 Routing Algorithm Approach

To deal with the routing problem, we applied two optimization strategies based on computational intelligence: genetic algorithm and particle swarm optimization. The implementation of the routing algorithm requires the definition of essential parameters. One of them is the definition of the solution.

Each solution contains a possible path for a source-destination pair. The latter is selected at random from an array of several paths of the same source-destination, where this array contains several paths of each pair (source-destination). Figure 3 shows an example of several paths of such pairs and how to define a solution.

Therefore, a solution contains a set of possible paths for all the source-destination pairs needed to transmit data, based on the application mapped onto the NoC. Figure 4 shows a solution. In order to measure the quality of the solutions, we evaluate the fitness function. This function will make it possible to distinguish the best solutions. Therefore, the best solution is the least congested, compared to others.

The fitness function is the congestion. It can be considered as a cost, that we have defined to estimate the rate of use of the network links. Initially, the congestion cost of

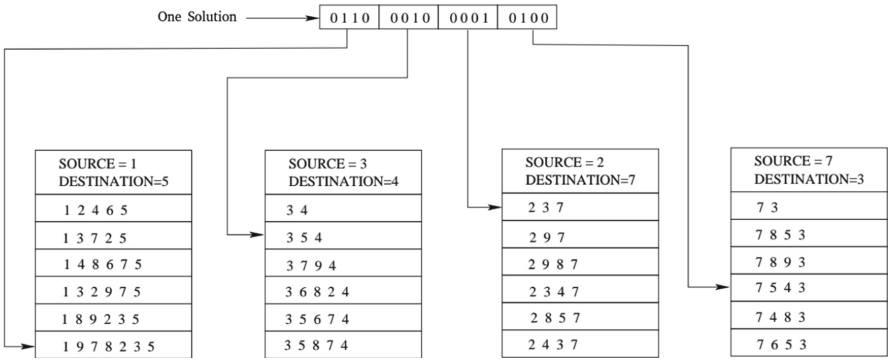


Fig. 3. Example of several paths of pairs (source-destination)

CHROMOSOME	0110 0010 0001 0100
Route 1	1 9 7 8 2 3 5
Route 2	3 5 4
Route 3	2 3 7
Route 4	7 5 4 3

Fig. 4. Exemple of one solution

all links is equal to 0. Each time a link appears in a path, its cost is incremented by 1. The congestion cost of a path of length L is given by the Eq. 1:

$$Congestion_path_i = \sum_{j=1}^{L} C_j, \tag{1}$$

where C_j is the congestion cost associated with the j-th link of path i in the routing table [17]. The total congestion cost of a solution is the sum of all congestions of all paths used in the solution and calculated according to the Eq. 2:

$$Congestion(S) = \sum_{i=1}^{N} Congestion_path_i. \tag{2}$$

3.1 Genetic Algorithm

Genetic algorithm (GA) is a special type of stochastic research algorithm that applies evolution as a problem-solving technique [18]. GA works on the research space called population [19]. Each element of the population is called a chromosome, where each one is a solution in itself.

At the beginning, a random population is generated, which represents possible solutions. Each solution is represented by an array, in which the size is the number of source

and destination node pairs present in the system. For each pair, a path is selected from multiple distinct paths existing between source and destination.

Once the initial population is generated, a new one is generated based on the following genetic operations [20]:

1) Selection: consists in choosing the individuals who will participate in the reproduction of the future population. The selection function may be random or according to a probabilistic method, such as roulette and tournament. An individual is chosen so that it participates in the reproduction stage to form the new population. In this work, we use random selection.

2) Crossover: recombines two 'parents' to produce two new 'children' in the next generation. We used one-point crossing, where two parent paths are split into two parts, respectively, and recombined. The parts are exchanged to produce the new children. The bit crossing positions are randomly selected along the length of the chromosome, as shown in Fig. 5.

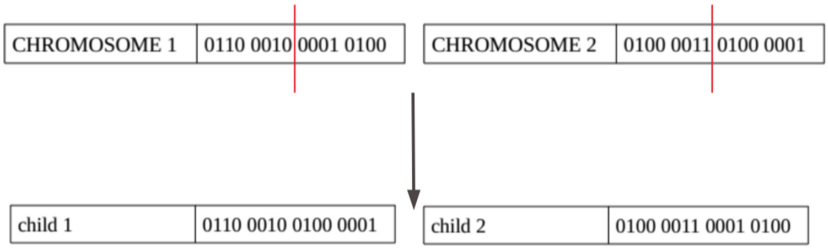


Fig. 5. Crossover operation

3) Mutation: randomly flips a bit position in the chromosome. It consists of choosing a gene at random and replace it with another value, chosen randomly, as shown in the Fig. 6. In the mutation, a mutation rate must be applied which refers to the probability or frequency with which a gene in the chromosome of a solution undergoes a random change or mutation during the evolutionary process.

Algorithm 1 shows the main steps of the genetic algorithm. The stop criterion is the maximum number of iterations, which relates to the number of generations. We applied 500 iterations, a population of 100 chromosomes and a mutation rate of 0.9.

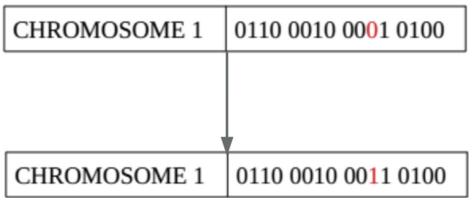


Fig. 6. Mutation operation

Algorithm 1. The main steps of Genetic Algorithm

```
Initialize the individuals of the population
Evaluate and initialize best solution
iteration := 0
while iteration < max_iteration do
    (a) Selection()
    (b) Crossover()
    (c) Mutation()
    iteration := iteration + 1
end while
```

3.2 Particle Swarm Optimization Algorithm

Particle Swarm Optimization (PSO) [21] is one of the examples of swarm intelligence strategies inspired by a flock of birds in search for food. In PSO, each particle position represents a potential solution of the problem. Each particle is characterized by its position and its performance, i.e., in the case of a function optimization, each particle is characterized by the variable value and the corresponding function value.

Particles are distributed randomly in the search space and move according to a local information. Each particle informs its performance to the neighbouring ones, can remember the position where registered its best perfomance so far and can inform the position of the neightbouring particle with the best performance. The new position x_i of particle i is given by Eq. 3:

$$x_i(t+1) = x_i(t) + v_i(t+1). \tag{3}$$

The velocity v_i of each particle i is given by Eq. 4:

$$v_i(t{+}1) = w.v_i(t){+}c_1.r_1(t{+}1)(Pbest(t){-}x_i(t)){+}c_2r_2(t{+}1)(Gbest(t){-}x_i(t)), \tag{4}$$

where:

(i) w is the inertia coefficient, used to prevent the particle from drastic changes in direction;

(ii) c_1 is the cognitive coefficient, related to the performance of particle i with respect to its previous one;
(iii) c_2 is the social coefficient, related to the performance of parcicle i with respect to the performance of the swarm;
(iv) $r_1(t)$ and $r_2(t)$ are random values in the interval [0,1], used to characterize the stochastic nature of the cognitive and social contributions;
(v) $Pbest(t)$ is the best position founded by particle i so far;
(vi) $Gbest(t)$ is the best position founded by the swarm.

The cognitive coefficient c_1 and the social coefficient c_2 yield a better performance when they are balanced [23].

In our work, the position of a particle is an array, whose size is the number of pairs (source, destination) implemented in the NoC. For each pair (source, destination), there will be several different paths between the source and the destination. The particle is chosen from each pair among the ones that exist.

Algorithm 2 shows the main steps of the PSO. The stop criterion is the maximum number of iterations. We applied 500 iterations, a swarm of 200 particles, the cognitive coefficient and the social coefficient are 1.35, each, and the inertia coefficient is 0.8.

Algorithm 2. The main steps of PSO

```
Initialize the particles of the swarm
Evaluate all the particles of the swarm and save Gbest
Initialize the velocity for each particle
iteration := 0
while iteration < max_iteration do
  for each particle do
    Select a Pbest
    Update the position using Eqs. 3 and 4
    Evaluate the particle fitness
    Update Pbest
  end for
  Update Gbest
  iteration := iteration + 1
end while
Return result from Gbest
```

4 Simulations and Performance Results

We used the Access-Noxim simulator [24], which is an upgraded version of the Noxim simulator. Noxim is a NoC simulator tool [25] written in C++ programming language, based on the SystemC library. The implemented routing strategy can be modified easily by writing and testing code using C++.

In this work, we focus on latency as a measure of performance. Latency is the time between the packet injection in the local port and the consumption of the queue flit in the local port of the destination node. The network was simulated with both 2D and 3D NoCs. We used a 5×5 mesh, for the 2D NoC, and a $3 \times 3 \times 3$ mesh, for the 3D NoC. The total simulation time was 20000 cycles. All the tests were carried out varying the routing algorithms, the traffic models and the packet injection rate (PIR).

Source-destination pairs are generated according to three different distribution traffic models: Complement, Transpose 1 and Transpose 2 [26]. These models are called deterministic patterns. In the three patterns, each node sends messages to a single destination and the destination nodes are selected based on the position of the source nodes, as seen in Table 1. We compared the results obtained with DOR (XYZ) and Odd-Even algorithms. Table 2 shows the full system simulation configuration.

Table 1. 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)

Table 2. Parameter settings

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

Figure 7 reports the average latency as a function of PIR for the 2D NoC, using Complement in Fig. 7(a), Transpose 1 in Fig. 7(b) and Transpose 2 in Fig. 7(c). When PIR is very low, all algorithms perform in the same way and have approximately the same average latency values. When PIR is high, GA and PSO outperform the other routing strategies and provide the best results compared to XYZ and Odd-Even.

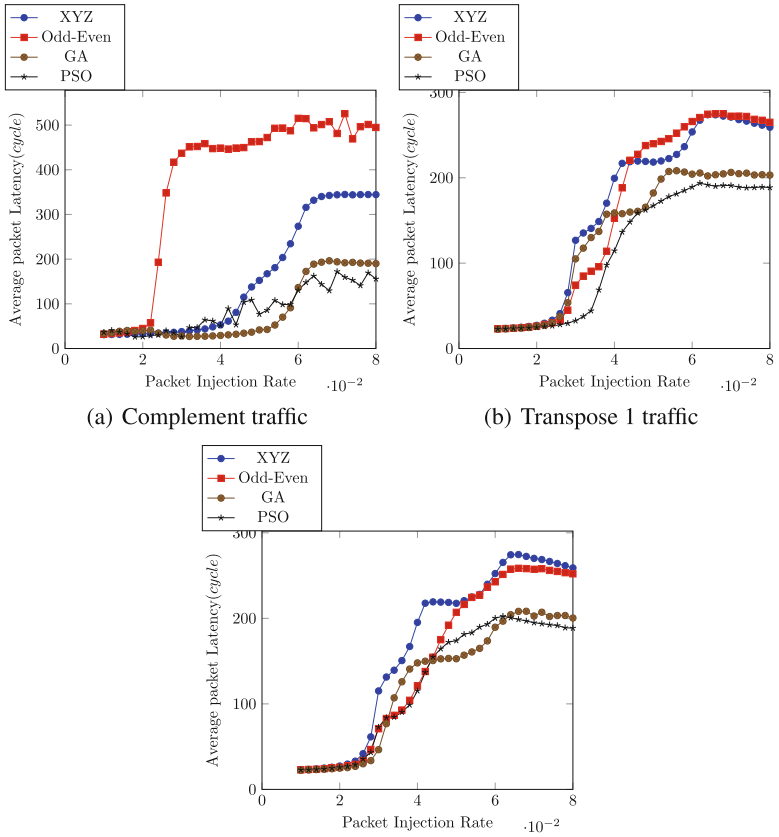


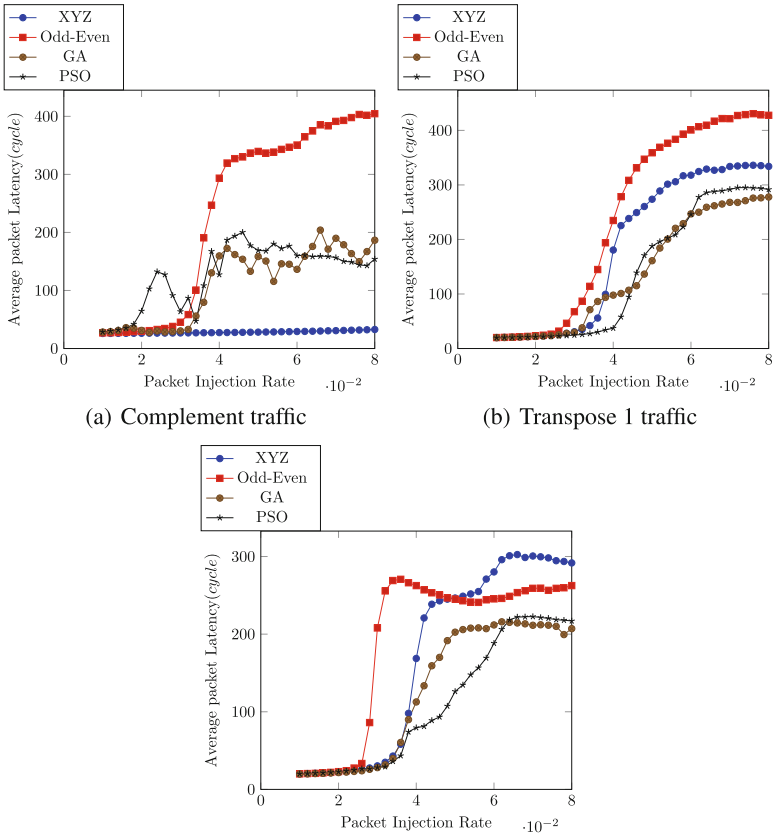
Fig. 7. Average latency as a function of PIR in 2D NoC

In Complement traffic, the maximum load, before the network starts saturating using Odd-Even routing, is 0.025 packets/node/cycle. This value increases to 0.045 with XYZ. GA outperforms the other algorithms and PSO starts to give low latency, compared to GA, when PIR it reaches 0.06.

In Transpose 1, the network starts saturating at 0.0135 and 0.015 for Odd-Even and XYZ, respectively. PSO achieves better performance, while GA starts to provide low latency when PIR reaches 0.047, compared to PSO.

Finally, for Transpose 2, XYZ starts saturating at 0.02. When GA and PSO reach 0.01 PIR, the latency becomes better than Odd-Even. PSO begins to provide low latency when it reaches 0.065.

Figure 8 reports the average latency as a function of PIR in the 3D NoC. Figure 8(a) relates to Complement traffic, Fig. 8(b) to Transpose 1 traffic and Fig. 8(c) to Transpose 2 traffic.



(a) Complement traffic (b) Transpose 1 traffic

(c) Transpose 2 traffic

Fig. 8. Average latency as a function of PIR in 3D NoC

In Complement traffic, XYZ outperforms the other routing algorithms in terms of average latency because, in the complementary model, the paths taken by the XYZ routing are static and do not cross in most parts, which does not cause congestion. PSO and GA outperform Odd-Even, when PIR starts saturating at 0.03, while begins to give better performance when it reaches 0.06.

In Transpose1 traffic, 3D Odd-Even performs poorly with regard to XYZ, PSO and GA, and reaches a saturation point at 0.026. The network saturates at 0.046 for XYZ. PSO and GA provide better results, while GA outperforms PSO when PIR is 0.06.

Finally, in Transpose 2 traffic, the network saturation points are 0.046 and 0.042 for Odd-Even and XYZ, respectively. GA and PSO achieve better performance than Odd-Even and XYZ when PIR is 0.04, while PSO outperforms GA when PIR reaches 0.03.

5 Conclusion

This work proposes a routing algorithm based on genetic algorithm and particle swarm optimization, for Network-on-chip in a 3D mesh topology, in order to minimize congestion. A large number of tests were performed, through simulation, using deterministic generated traffic. In most cases, both algorithms performed better than other routing algorithms, such as XYZ and Odd-Even. Despite the promising results that were obtained in deterministic traffic, we have some work to do in order to improvement the approach. In future work, we will use other computational intelligence meta-heuristics, such as differential evolution, and graphs of real application tasks. Besides this, we intend to experiment them on NoCs with larger dimensions than the ones used here, in order to address more intense and complex communication traffics.

Acknowledgments. The authors acknowledge the financial support of Fundação Carlos Chagas Filho de Amparo à Pesquisa do Estado do Rio de Janeiro - FAPERJ - (Proc. E-26/210.044/2021)

References

1. 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)
2. Davis, W.R., et al.: Demystifying 3D ICs: the pros and cons of going vertical. IEEE Des. Test Comput. **22**(6), 498–510 (2005)
3. Bougherara, M., Nedjah, N., Mourelle, L.D.M., Rahmoun, R., Sadok, A., Bennouar, D.: IP assignment for efficient NoC-based system design using multi-objective particle swarm optimisation. Int. J. Bio-Inspired Comput. **12**(4), 203–213 (2018)
4. 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
5. Sullivan, H., Bashkow, T.R.: A large scale, homogeneous, fully distributed parallel machine, i. In: ACM SIGARCH Computer Architecture News, vol. 5, no. 7, pp. 105–117. ACM (1977)
6. Valiant, L.G., Brebner, G.J.: Universal schemes for parallel communication. In: Proceedings of the Thirteenth Annual ACM Symposium on Theory of Computing, pp. 263–277. ACM (1981)
7. Seo, D., Ali, A., Lim, W.T., Rafique, N.: Near-optimal worst-case throughput routing for two-dimensional mesh networks. In: ACM SIGARCH Computer Architecture News, vol. 33, no. 2, pp. 432–443. IEEE Computer Society (2005)
8. Chiu, G.M.: The odd-even turn model for adaptive routing. IEEE Trans. Parallel Distrib. Syst. **11**(7), 729–738 (2000)
9. 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)
10. Ebrahimi, M.: Fully adaptive routing algorithms and regionbased approaches for two-dimensional and three-dimensional networkson-chip. lET Comput. Digit. Tech. **7**(6), 264–273 (2013)
11. 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)

12. 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)
13. Jouybari, H.N., Mohammadi, K.: A low overhead, fault tolerant and congestion aware routing algorithm for 3D mesh-based Network-on-Chips. Microprocess. Microsyst. **38**(8), 991–999 (2014)
14. Junior, L.S., 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)
15. Silva Junior, L., Nedjah, N., De Macedo Mourelle, L.: Efficient routing in network-on-chip for 3D topologies. Int. J. Electron. **102**(10), 1695–1712 (2015)
16. Alfaraj, N., Zhang, J., Xu, Y., Chao, H.J.: Hope: hotspot congestion control for clos network on chip. In: 2011 Fifth IEEE/ACM International Symposium on Networks on Chip (NoCS), pp. 17–24 (2011)
17. Nychis, G.P., Fallin, C., Moscibroda, T., Mutlu, O., Seshan, S.: On-chip networks from a networking perspective: congestion and scalability in many-core interconnects. In: Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, SIGCOMM 2012, ACM, New York, NY, USA, pp. 407–418 (2012)
18. Holland, J.H.: Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence. MIT Press, Cambridge, MA, USA (1992)
19. Goldberg, D., Holland, J.: Genetic algorithms and machine learning (1988)
20. Mohammed, M.A., Ahmad, M.S., Mostafa, S.A.: Using genetic algorithm in implementing capacitated vehicle routing problem. In: 2012 International Conference on Computer & Information Science (ICCIS), pp. 257-262. IEEE (2012)
21. Zhang, L.B., Zhou, C.G., Liu, X.H., Ma, Z.Q., Ma, M., Liang, Y.C.: Solving multi objective optimization problems using particle swarm optimization. In: Congress on Evolutionary Computation (CEC 2003), vol. 3, pp. 2400–2405 (2003)
22. 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
23. Nedjah, N., Mourelle, L.D.M.: Evolutionary multi-objective optimisation: a survey. Int. J. Bio-Inspired Comput. **7**(1), 1–25 (2015)
24. 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 (2015)
25. Access Noxim. http://access.ee.ntu.edu.tw/noxim/index.html
26. Duato, J., Yalamanchili, S., Ni, L.: Interconnection Networks. Morgan Kaufmann, Burlington (2002)