IAES International Journal of Robotics and Automation (IJRA)
Vol. 15, No. 2, June 2026, pp. 488∼502
ISSN: 2722-2586, DOI: 10.11591/ijra.v15i2.pp488-502

Optimized mapping in 2D and 3D network on chip using Bat algorithm

Maamar Bougherara[1,2], Rafik Amara[1,3], Amina Guidoum[1]
[1]Department of Computer Science, High Normal School of Kouba, Algiers, Algeria
[2]Lim Laboratory, Faculty of Exact Sciences, Akli Mohand Oulhadj University of Bouira, Algeria
[3]LTIR Laboratory, Faculty of Electronics and Computational Science, USTHB University, Algiers, Algeria

Article Info

Article history:

Received Aug 17, 2025
Revised May 6, 2026
Accepted May 13, 2026

Keywords:

Bat algorithm
Communication cost
Mapping
Network on chip
Optimization

ABSTRACT

Communication within system-on-chip (SoC) architectures has evolved significantly to keep pace with the growing complexity of modern applications. To overcome the limitations of traditional interconnects, network-on-chip (NoC) has emerged as a scalable and efficient communication solution. Although early NoC designs relied heavily on 2D architectures, their physical and performance constraints have led to the rise of 3D NoC architectures, which offer better spatial integration and improved performance. In order to automate the NoC design process, a number of electronic design automation (EDA) tools and optimization algorithms are employed to help designers achieve efficient and high-performance designs. Within this EDA framework, one of the most critical stages is the core placement or application mapping phase, where computational tasks are allocated to the processing elements of the architecture. This step is very hard due to its combinatorial nature, and its optimization is essential since it directly impacts communication cost, energy consumption, and overall system performance. To address this challenge, numerous heuristic and metaheuristic algorithms have been explored for both 2D and 3D NoCs. In this paper, we propose an adaptation of the bat algorithm to solve the mapping problem in both 2D and 3D NoC architectures, with the objective of minimizing communication cost. The proposed approach is evaluated and compared against other optimization methods to assess its effectiveness in enhancing NoC performance within the EDA framework.

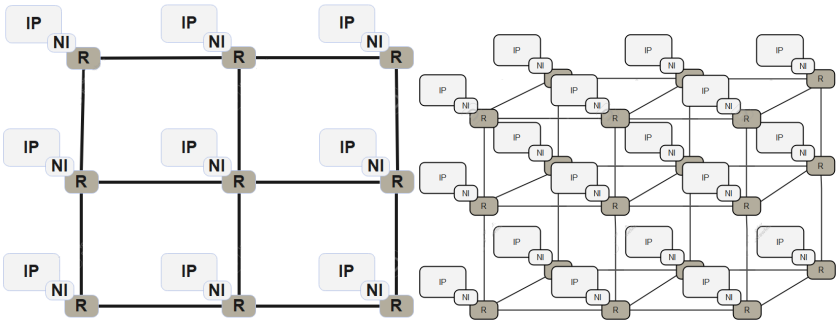
Corresponding Author:

Maamar Bougherara
Department of Computer Science, High Normal School of Kouba
Algiers, Algeria
Email: bougherara.maamar@gmail.com

1. INTRODUCTION

As the demand for high-performance computing systems continues to grow, network-on-chip (NoC) architectures have emerged as a promising solution to address communication bottlenecks in complex system-on-chip (SoC) designs [1]. Similar in concept to traditional computer networks, a NoC provides scalable and structured communication among multiple processing elements. However, due to stringent constraints in area, power, and performance, NoC designs must be carefully optimized to meet the specific requirements of their target applications. In NoC-based systems, applications are typically divided into several computational tasks, each encapsulated in an intellectual property (IP) block. These IPs must be integrated into the NoC infrastructure in a manner that maximizes performance while minimizing energy consumption and latency [2].

A typical NoC consists of four key components [3]: i) IP cores, which include processors, memory units, and custom controllers; ii) network interfaces (NIs), which serve as a bridge between the IP cores and NoC communication; iii) routers, which handle packet forwarding and arbitration; and iv) physical interconnects, which define the paths for data transmission. Each tile consists of an IP, a network interface (NI), and a router. The topology of a NoC determines how tiles are connected [4]. Among the available configurations, the 2D mesh topology is the most widely adopted due to its regular structure, simplicity, and scalability. With the increasing complexity and integration density of modern SoCs, 2D NoC architectures face growing limitations in terms of bandwidth, latency, and energy efficiency. To address these challenges, three-dimensional (3D) NoC architectures have been proposed [2]. By vertically stacking multiple 2D layers and employing through-silicon via (TSV) technology [2], 3D NoCs significantly reduce the average communication distance, enhance bandwidth, and improve performance. In a 3D mesh topology, each router connects to up to six neighboring routers [5], four in the same layer and two in adjacent vertical layers. This configuration enables both intra-layer and inter-layer communication, leading to better parallelism, reduced latency, and higher throughput. Figure 1 shows 2D and 3D mesh NoC topologies. The design of a NoC-based system generally follows three main stages [6]: i) task assignment, which involves assigning application tasks to a predefined library of IP blocks; ii) IP mapping, where these IP blocks are mapped to physical nodes (routers) within the NoC; and iii) static routing, which determines the communication paths between the mapped IPs. Each of these stages is supported by EDA tools, which explore the design space to generate optimized hardware/software configurations. Figure 2 illustrates a standard embedded system design flow for NoC platforms. This paper introduces a novel model for solving the IP mapping problem in both 2D and 3D NoCs. After modeling the problem, defining the main objective of the study, and formulating the cost function, the Bat algorith is employed for the first time as a new method to address the application mapping problem.



(a) 2D (b) 3D

Figure 1. Mesh-based NoC

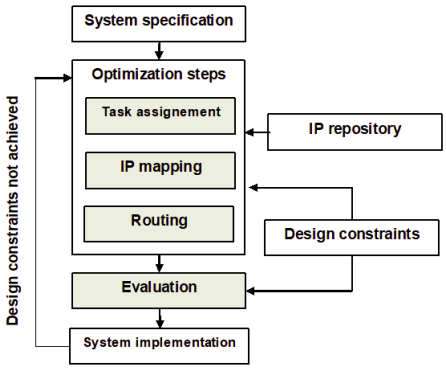


Figure 2. Typical embedded system design flow for NoC platform.

The structure of this paper is as follows. Section 2 introduces the NoC generalities. Section 3 provides a review of prior research related to mapping in NoC architectures. In section 4, we present the modeling of

the mapping problem prior to applying a metaheuristic approach. The bat algorithm, used as the core mapping strategy, is described in section 5. Section 6 analyzes the results obtained from the simulation experiments. Finally, section 7 presents the conclusion of the study along with directions for future work.

2. NOCS GENERALITIE

NoC draws its inspiration from communication networks originally designed for supercomputers and is formed by interconnected on-chip components that communicate through packet-based transmission over a scalable interconnection architecture. NoC offers numerous benefits, including energy efficiency, reliability, bandwidth scalability in comparison to traditional bus architectures, and reusability [6]. A NoC's topology refers to the arrangement of its components within the on-chip interconnect, which can be structured as a ring, torus, or 2D mesh. It is also characterized by other aspects such as the communication mode, flow control mechanisms to prevent deadlocks, and buffering policies. Several steps are required in the design of a NoC based system. Initially, the application is decomposed into a set of parallel communication tasks. Then, each task is assigned to a selected processing core, which is scheduled according to the system requirements. Finally, the processing cores are mapped onto the NoC architecture [5]. This paper focuses on the application mapping stage, which remains an open research problem. An optimal mapping can achieve up to 51.7% communication energy savings compared to an ad hoc implementation [7]. To achieve high performance, an optimal mapping must be determined. Given m tasks mapped onto a NoC with n cores, where $m \leq n$, the number of possible mappings is $n!/(nm)!$. As an NP-hard combinatorial optimization problem, application mapping is generally addressed using heuristic algorithms to obtain suboptimal solutions. In conventional 2D NoCs, each IP is connected to a router via a network interface, and these routers are arranged in a planar grid with wired links. Communication takes place using packet-switched data transfer, where messages are broken into packets and routed through the NoC [8]. Compared to traditional bus-based systems, 2D NoCs offer several advantages: higher communication bandwidth, reduced latency, improved scalability, and lower power consumption [9]. However, as more IPs are integrated into a single chip, the average number of hops (routers traversed by a packet) increases, leading to greater communication energy and degraded performance. This issue becomes increasingly problematic as system size grows [7]. The 3D NoC paradigm addresses these issues by enabling vertical communication through TSVs. This approach reduces the average hop count and significantly lowers latency and energy consumption. Additionally, it enhances area utilization, making it suitable for large-scale multicore systems. Consequently, 3D NoCs have gained significant interest in both academia and industry as a compelling solution for future high-performance computing platforms. This work focuses on optimizing the mapping phases of NoC design, aiming to minimize communication energy while maintaining high system performance.

In a 3D NoC, the IP mapping problem consists of assigning each IP core to a specific node within the 3D topology. This phase is critical, as efficient mapping can greatly reduce communication cost, energy usage, and latency [10]. Due to its combinatorial nature, the IP mapping problem is considered NP-hard and is closely related to the quadratic assignment problem (QAP). This complexity makes exact solutions impractical for large systems, especially within limited design time. As 3D NoCs enable the integration of a greater number of IPs across multiple layers using TSVs, efficient and scalable mapping algorithms are becoming increasingly important. To this end, heuristic and metaheuristic approaches offer a viable alternative, providing near-optimal solutions within reasonable time frames.

3. RELATED WORK

Several approaches have been proposed to the mapping in NoCs. These mapping techniques can be classified into three principal categories: exact approaches and heuristic (or approximate) techniques [11]. We show in this paper the most cited who take into a count single objective. Exact mapping methods, such as exhaustive search, linear programming, and branch-and-bound algorithms, rely on mathematical rigor to guarantee globally optimal solutions. An integer linear programming (ILP) formulation for the IP mapping problem is presented in [12]. This approach models both objectives and constraints as linear functions, with the additional requirement that decision variables take integer values. The ILP model is solved using the optimization tool, enabling precise solution computation within a mathematically defined space. Despite their accuracy, exact methods suffer from high computational complexity and significant runtime requirements. As a result, they are typically feasible only for small-scale mapping scenarios involving a limited number of IP

cores, where the size of the solution space remains manageable under current computing capabilities.

The second category encompasses heuristic-based approaches, which aim to provide efficient and practical solutions to the application mapping problem on NoC architectures. The NMAP algorithm [13] is a widely used heuristic that maps application cores to tiles iteratively. At each step, a core is selected and assigned to a tile, repeating the process until all cores are mapped. While NMAP includes an iterative improvement mechanism, the quality of the final solutions often remains constrained by the initial mapping. BMAP [14] introduces a binomial mapping approach based on iterative two-way merging, taking into account the traffic load between cores to optimize communication. CastNet [15] enhances solution diversity by leveraging the symmetric characteristics of mesh topologies. It selects multiple initial tiles and generates several mapping options, ultimately choosing the best one based on the number of free neighboring tiles for each core. CHMAP [16] calculates a priority value for each core based on its communication requirements and its position within a communication spanning tree. The algorithm then determines the mapping order and assigns cores to appropriate tiles according to these computed priorities. Onyx [17], proposed in 2009, introduces a lozenge-shaped mapping path and defines four movement patterns to assign priorities to tiles. This method has shown improved communication cost compared to earlier heuristics. Spiral [18] starts by mapping the highest-priority task at the center of the mesh, then progressively maps remaining tasks outward in a spiral pattern from already mapped cores. To address hierarchical communication, Dageleh and Jamali [19] proposed V-CastNet3D, a clustering-based mapping algorithm. Cores are grouped into clusters to reduce inter-cluster communication, and the CastNet heuristic is then applied within each cluster for mapping onto a 2D NoC mesh. For 3D meshes, certain techniques developed for 2D meshes have been adapted. For example, NMAP 3D, ILP3D, and PSMAP3D. Furthermore, CastNet3D, also proposed by Nalci *et al.* [12], extends this approach to 3D NoC architectures by optimizing the utilization of vertical links, thereby improving communication efficiency and reducing energy consumption.

Given the high computational cost of exact methods and the limitations of current processing capabilities, metaheuristic techniques have emerged as practical and scalable alternatives for solving the IP mapping problem in NoC architectures. Drawing inspiration from natural and biological phenomena, these approaches aim to efficiently explore large solution spaces and produce near-optimal solutions within reasonable computational time. In the context of 2D NoC architectures, several metaheuristic techniques have been proposed: In [20], CGMAP, a genetic algorithm-based approach, replaces the traditional random initialization with a chaotic mapping operator, enhancing the exploration capabilities of the algorithm. GBMAP [21] employs an evolutionary algorithm to efficiently map processing cores onto NoC tiles. Swarm intelligence-based approaches have also been explored. For example, a discrete multi-objective particle swarm optimization (PSO) method is proposed in [22], utilizing deterministic initial solutions to improve performance. In [23], a hybrid approach combining PSO and genetic algorithms further enhances solution quality. Ant colony optimization (ACO) algorithm is used to minimize bandwidth requirements, while a hybrid ACO-GA method is proposed in [24] to reduce communication costs. An artificial bee colony (ABC) algorithm is presented, where genetic operators are introduced with ABC in [25] to improve the communication-aware mapping efficiency. Cat swarm optimization (CSO) [26] is applied to map IPs onto 2D NoCs with notable results. Differential evolution (DE) is employed in [27] to address communication cost optimization in 2D NoC mapping problems.

While these techniques are primarily applied to 2D NoCs, recent research has extended metaheuristic strategies to 3D NoC architectures: In [28], a PSO-based algorithm is developed for mapping IPs onto a partially vertically-connected 3D mesh NoC. A PSO based mapping method is introduced in [22] to improve communication costs by applying bandwidth limitation. A similar approach is adopted in [29], but with the objective of minimizing energy consumption. Hang *et al.* in [28] apply quantum particle swarm optimization (QPSO) to the 3D NoC mapping problem, thereby reducing power consumption. An adaptive genetic algorithm based on logistic functions (LFAGA) is proposed in [30] to optimize the energy consumption of the network. Finally, a method based on fuzzy logic is proposed in [31]. An ABC-based method for 3D NoC mapping is introduced in [32], demonstrating improved results in communication optimization. A thermal-aware mapping technique is proposed in [33], which integrates genetic algorithms with fuzzy logic to minimize the peak temperature in 3D NoC systems. Finally, in [34], simulated annealing is employed to enhance communication efficiency in 3D NoC mappings.

Since our paper explores the application of the Bat algorithm (BA), it is important to highlight its prior use in the NoC design domain. In [35], the algorithm was applied to the routing phase, which follows the generation of an optimized mapping. In [36], the Bat algorithm was employed for the first time in a 3D

NoC architecture, but the study was restricted to small-scale real benchmarks with a limited number of cores. Similarly, in [37], BA was applied solely to the mapping problem in 2D NoC systems. In [38], BA was applied to the dynamic mapping problem in 2D NoC systems, where several faulty cores were introduced as a simulation scenario.

In contrast, the present work proposes a comprehensive Bat algorithm–based mapping strategy for both 2D and 3D NoC architectures. Extensive experiments were performed using real benchmarks and synthetic applications generated via TGFF, enabling a thorough analysis of the algorithm's scalability and performance as the number of cores increases. The obtained results clearly demonstrate the efficiency and robustness of the proposed approach when compared to other state-of-the-art optimization algorithms.

4. APPLICATION MAPPING PROBLEM

In the context of mapping problem in NoC, the application is typically modeled as an IP core graph, while the NoC infrastructure is represented as a topology graph. The IP mapping problem consists of assigning logical IP cores defined by specific communication relationships and bandwidth demands within the IP core graph to physical resource nodes in the NoC topology graph [8].

4.1. NoC topology graph

The topology of a NoC is abstracted as a directed graph $T(R, L)$, where: R is the set of nodes, with each node $r_k \in R$ representing a tile in the NoC. L is the set of directed edges, where each edge $l_{k,l} \in L$ represents a physical communication link between tiles r_k and r_l [26].

4.2. IP core graph

Similarly to NoC topology, the IP core graph is a high-level abstraction that represents the behavior of an application. It is denoted as a directed graph $G(C, A)$, where: C is the set of nodes, each node $c_i \in C$ representing an IP core, and A is the set of directed edges, where each edge $a_{i,j} \in A$ represents communication from IP source core c_j to destination core c_i [23]. Each edge $a_{i,j} \in A$ is associated with a weight $w_{i,j}$, which denotes the bandwidth requirement between the IPs c_j and c_i.

4.3. Mapping function in NoC

The mapping function defines the mapping of IP cores from the IP core graph $G(C, A)$ to nodes in the NoC topology graph $T(R, L)$, with the objective of minimizing communication energy [27]. For each IP core $c_i \in C$, there exists a corresponding node $r_k \in R$ in the NoC topology such that: map: $V \rightarrow T \quad map(c_i) = r_k, \forall c_i \in C \; \exists r_j \in R$ Furthermore, each IP core must be mapped to one node, ensuring that for any two IPs c_i and c_j: map(c$_i$) $\neq map(c_j)$

To satisfy this constraint, the number of nodes in the NoC topology graph must be greater than or equal to the number of nodes in the IP core graph. If necessary, dummy nodes can be added to the IP core graph to equalize the sizes of both graphs [25]. These dummy nodes are non-communicating placeholders and do not participate in any data exchanges.

4.4. Solution representation

The mapping solution is represented as a sequence with a permutation of the integers from 1 to n, where n is the number of nodes in the NoC topology graph [26]. Each sequence position represents a node in the NoC topology, and its value denotes the ID of the assigned IP core. An example mapping solution is shown in Table 1.

Table 1. Solution example

core number	1	2	3	4	5	6	7	8	9
node place	8	3	2	1	9	5	6	7	4

4.5. Communication cost

The main objective of this paper is to reduce communication cost by minimizing the number of hops for each communication when two tasks are mapped onto the NoC. The total communication cost is calculated using (1) [34].

$$commcost = \sum_i \sum_j w_{i,j} * nbhops(r_k, r_l), \tag{1}$$

where $c_i = source$, $c_j = distination$, $map(c_i) = r_k$, $map(c_j) = r_l$, $c_i \neq c_j$, and $r_k \neq r_l$. $nbhops()$ is the number of hops between the source and destination, calculated using the Manhattan distance, defined by the following function:

$$Hops(r_k, r_l) = |X_k - X_l| + |Y_k - Y_l| + |Z_k - Z_l| \quad (2)$$

where (X_k, Y_k, Z_k) and (X_l, Y_l, Z_l) represent the coordinates of the nodes r_k and r_l within a 3D mesh NoC, respectively. For a 2D mesh NoC, Z_k and Z_l are equal to 0.

5. APPLICATION MAPPING WITH BAT ALGORITHM

5.1. Bat in nature

Microbats, a subgroup of bats, are the only mammals capable of sustained, active flight. They possess an extraordinary ability known as echolocation, which enables them to navigate and hunt efficiently in complete darkness. To detect and locate their prey, microbats emit high frequency ultrasonic pulses typically ranging from 25 to 150 kHz [39]. Each short pulse, lasting approximately 5 to 20 milliseconds, generates an echo upon striking an object or prey. By analyzing these echoes, the bat can precisely determine the distance, direction, and even the size of the target.

During the initial search phase, a bat emits an average of 10 to 20 pulses per second. However, as it closes in on its prey, it dramatically increases the pulse rate up to 200 pulses per second—while simultaneously reducing the intensity of its emitted signals [40]. This adaptive strategy helps avoid echo overlap and enhances localization accuracy. The bat's flight path adjusts continuously in response to real time acoustic feedback, allowing for agile and precise movements.

This remarkable natural behavior—balancing global exploration when the prey is distant with fine tuned exploitation as it nears the target has inspired the development of a nature-inspired optimization technique known as the Bat algorithm (BA). Initially proposed to address complex global optimization problems, the BA mimics key aspects of echolocation: frequency variation, loudness modulation, and dynamic search adaptation[41]. BA has demonstrated strong performance across various technical domains, including multi objective optimization, machine learning, and scheduling problems. Its strength lies in its ability to seamlessly combine broad exploration of the solution space with intensive local refinement, mirroring the intelligent hunting strategies of bats in the wild[42].

5.2. Bat algorithm steps

The Bat algorithm, a nature inspired optimization method, was proposed by Yang in 2010 [39]. It is based on the echolocation behavior of microbats, which use high-frequency sound pulses to locate prey and avoid obstacles especially during twilight hours. This natural mechanism was mathematically modeled to create a novel and efficient optimization technique that simulates the dynamic and adaptive flight patterns of bats in the wild. This section presents the steps of the Bat algorithm, inspired by the real-life behavior of bats in nature. To define the algorithm, Yang introduced six key steps, which are outlined below [40].

a. Step 1: Initialization of Bat population and problem parameters

Consider a d-dimensional search space represented by a population of bats. The total number of bats (flock size) is denoted by N. The position of each bat i at iteration t is expressed as a vector in this d-dimensional space [41].

$$X_i^t = [x_{i,1}, x_{i,2}, x_{i,3} \ldots x_{i,d}] \quad (3)$$

The Bat algorithm begins by randomly initializing a population of virtual bats and setting the parameters required for the optimization problem. The objective is to evaluate the quality of each candidate solution x by computing the objective function $f(x)$, as defined in the problem.

b. Step 2: Bat population memory initialization

Each bat i is equipped with a memory that stores the location of its personal best position m_i. At iteration t, this corresponds to the best position found by bat i while exploring the search space. During the initialization stage, the position of each bat is randomly generated. Once generated, all solution vectors are stored in the bat memory [40]. At this point, the best global solution, denoted as X_{best}, is initialized and corresponds to the best bat position in the memory according to the objective function.

c. Step 3: Bat movement

In this step, each bat i flies at a speed v_i assigned to a randomly generated frequency f_i. The new position of bat i in the search space is updated as follows [41]:

$$V_i^{t+1} = V_i^t + (X_i^t - X_{best})f_i \tag{4}$$

$$X_i^{t+1} = X_i^t + V_i^{t+1} \tag{5}$$

$$f_i = f_{min} + (f_{min} - f_{max})\beta \tag{6}$$

where $\beta \in [0..1]$. The position of bat i is updated based on its previous position and increased by a relatively small value. This value is inherited as the distance between the global best position and the parent bat becomes close to the descendant bat.

d. Step 4: Intensification of the current bat population

This step introduces controlled randomness into the Bat algorithm through a local search mechanism. With a certain probability defined by the pulse rate R_i, each bat may perform a local random walk around the best-known solution in the population. A historically good solution m_i is selected from the current best individuals, and the new position of bat i is updated as follows [43]:

$$X_i^{t+1} = \begin{cases} m_i + \epsilon A^t & \text{if } rand > R_i \\ X_i^t + V_i^{t+1} & otherwise \end{cases} \tag{7}$$

here, ϵ is a random number drawn from a uniform distribution $[-1..1]$, and A^t represents the average loudness of all bats in the population at iteration t. This strategy allows bats to exploit promising areas of the search space while still maintaining some degree of randomness.

e. Step 5: Updating the bat population memory

For each bat in the population memory (BM), the newly generated solution $x_i^{'}$, $x_j^{'}$ may replace the current solution x_i, x_j under the following conditions [44]:

$$\begin{cases} f(X_i^{t+1}) < f(X_{best}) \\ Rand(0..1) < A_i^t \end{cases} \tag{8}$$

Similar to natural behavior, the loudness A_i and the pulse emission rate R_i are updated dynamically during the optimization process. As the iterations progress, A_i gradually decreases, whereas R_i increases according to [41]:

$$A_i^{t+1} = \alpha A_i^t \tag{9}$$

$$R_i^{t+1} = R_i^0(1 - e^{(-\sigma \times t)}) \tag{10}$$

where $R_i^t \to R_i^0$, $A_i^t \to 0$, and $t \to \infty$. R_i^0 is the maximum pulse emission rate. $\alpha \in [0..1]$ is the loudness damping factor, $\sigma > 0$ is the pulse rate increment factor, and t is the current iteration number. The parameter α is comparable to the cooling factor in simulated annealing.

f. Step 6: Termination criterion

The Bat algorithm repeats steps 3 through 5 until a termination condition is satisfied. Common stopping criteria include:

- Reaching a maximum number of generations,
- Exceeding a computational time limit,
- A fixed number of non-improving iterations,
- Achieving a desired objective value (solution quality threshold).

Once the termination condition is met, the algorithm halts and returns the best solution found during the search process. The details of the algorithm steps are presented in algorithm 1.

Algorithm 1. The Bat Algorithm

Generate the population of bats
Evaluate each bat using the objective function
Initialize the memory of each bat
Initialize the velocity of each bat
Initialize R_i^0 and A_i^0 for each bat
Initialize the frequency f_i of each bat using (6)
Save the best global solution
iter $\leftarrow$ 0
while iter $<$ maximum number of iterations **do**
  for each bat i **do**
    Update velocity and position using (4) and (5)
    Generate a random number $rand_1$ for use in (7)
    Evaluate the new position of the bat
    Generate a random number $rand_2$ for use in (8)
    Increase R_i and decrease A_i using (9) and (10)
    Update the memory of the bat
    Update the best global solution
  end for
  iter $\leftarrow$ iter + 1
end while
return the best global solution

5.3. Modeling Bat algorithm for mapping problem

In order to model the Bat algorithm so that it can effectively address the mapping problem, we have detailed its steps by aligning them with the specific requirements of the problem.

a. Step 1: Initialization of Bat population and problem parameters

At this stage, the parameters required for calibrating the Bat Algorithm are initialized by defining the optimization problem, objective function, constraints, and key algorithm settings to achieve optimal performance.

- N: number of bats (population size).
- $maxt\_iter$: maximum number of iterations.
- R_i^0: maximum pulse emission rate.
- A_i: loudness of bat i.
- σ and α: constants.
- f_{min} and f_{max}: minimum and maximum frequencies used by the bats.

b. Step 2: Initialization of Bat positions and memories

Randomly position N bats in a d-dimensional search space, where each bat represents a feasible solution to the placement problem according to (3). Each bat stores its best-found position in memory, which is initially set to its current position at iteration 0.

c. Step 3: Evaluation of initial fitness

For each bat, evaluate the quality of its current position by substituting the corresponding decision variable values into the objective function and save the global best solution.

d. Step 4: Generation of new positions

Each bat updates its position in the search space using its velocity and the global best solution according to (4) and (5). The feasibility of the newly generated position is evaluated based on (8).

e. Step 5: Feasibility checking of new positions

For each bat, the feasibility of the newly generated position is evaluated according to the problem constraints. If the position is feasible, the bat updates its current position accordingly.

f. Step 6: Parameter updating

In this step, each bat updates its memory and adjusts its parameters by increasing its pulse emission rate and decreasing its loudness according to (9) and (10).

g. Step 7: Stopping criterion checking

The loop comprising steps 5 to 6 is repeated until the predefined maximum number of iterations ($iter_{max}$) is reached. Once the stopping criterion is satisfied, the best solution stored in memory, in terms of the objective function value, is returned as the final solution to the optimization problem.

6. EXPERIMENT RESULT

To evaluate the performance of a NoC architecture, a variety of benchmarks are commonly utilized [17]. A benchmark typically consists of a set of interdependent tasks that emulate real application workloads. These tasks represent computational elements and the communication between them, allowing designers to evaluate architectural aspects such as energy efficiency and communication cost. In this study, two main categories of benchmarks are considered:

a. Real-world benchmarks

These benchmarks are derived from actual multimedia applications and are frequently used for evaluating the performance of NoC architectures. Figures 3(a) to 3(d) illustrates four widely adopted benchmarks, video object plane decoder (VOPD), moving picture experts group (MPEG4), picture in picture (PIP), multi-window display (MWD).

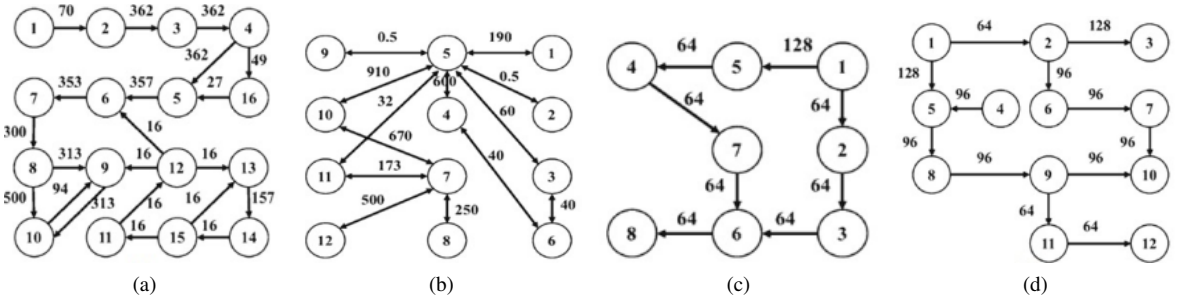


Figure 3. Benchmarks used in the experiments (a) VOPD – video object plane decoder, (b) MPEG4 – moving picture, (c) PIP – picture in picture, and (d) MWD – multi-window display experts group

b. Synthetic benchmarks

These benchmarks are automatically generated using the task graphs for free (TGFF) tool [45], which enables the creation of various task graphs with controlled properties such as task count, communication volume, and execution time. Synthetic benchmarks play an important role in evaluating the performance of NoC architectures under diverse scenarios. In this work, five benchmarks generated using TGFF and commonly adopted in NoC studies [46] are utilized for testing.

6.1. Setting

Before presenting the experimental results, we detail the configuration parameters used to calibrate the algorithm for optimal performance, as well as the 2D and 3D topologies employed (Tables X and Y). For real-world benchmarks, VOPD, MPEG4, and MWD were mapped onto a 4×4 2D NoC and a 2×4×2 3D architecture, while the smaller PIP benchmark was mapped onto a 2×2×2 3D topology. For synthetic benchmarks, TG0, TG1, and TG2 were mapped onto a 6×6 2D topology and a 3×3×3 3D topology, whereas TG3 and TG4 were mapped onto an 8×8 2D topology and a 4×4×4 3D topology. Algorithm calibration was performed through multiple tests, with the population size set to 500 and the maximum number of iterations set to 1500. The initial loudness A_i was set to 0.25, the pulse emission rate r_i to 0.25, and the frequency range was defined between 0 and 500.

6.2. Result and discussion

The results reported in Table 2 present the performance of the Bat mapping approach applied to both 2D and 3D NoC topologies across Synthetic benchmarks. and The graphical representations in Figure 4 allow for a comparison of the mapping results achieved by the Bat algorithm in 2D and 3D NoC.

The data clearly indicate that the 3D NoC configuration consistently outperforms its 2D counterpart, underscoring the benefits of exploiting the third dimension in task-to-core mapping. These results demonstrate that incorporating the additional spatial dimension not only optimizes resource allocation but also enables more efficient handling of complex application graphs, which is often challenging in traditional 2D NoCs.

Table 2. Comparison of obtained results

Benchmarks	# Task	# Edge	Bat_2D	Bat_3D
TG0	12	13	30300	18200
TG1	16	16	43600	34400
TG2	27	27	89000	61800
TG3	30	34	109400	109300
TG4	50	59	271700	201700

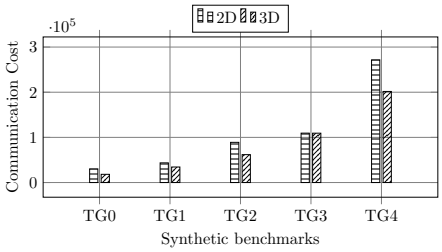


Figure 4. Comparison between 2D and 3D mapping

To thoroughly evaluate the effectiveness of our Bat-based mapping approach, we first compared it with improved versions of algorithms previously proposed by the same authors for 3D NoC architectures, namely ABC [25] and simulated annealing [34]. This initial comparison provides a measure of the competitiveness of our method against already optimized reference techniques. The results presented in Table 3 summarize the performance of the Bat mapping approach applied to 2D and 3D NoC topologies using synthetic benchmarks. Furthermore, the graphical illustrations in Figure 5 provide a clear comparison of the mapping outcomes obtained for both 2D and 3D NoCs, enabling a deeper analysis of the differences between the two architectures in comparison with Bat, SA, and ABC approaches.

Table 3. Comparison results in synthetic benchmarks

Benchmarks	SA_2D	ABC_2D	Bat_2D	SA_3D	ABC_3D	Bat_3D
TG0	32400	49300	30300	27800	31700	18200
TG1	39700	63400	43600	37200	40600	34400
TG2	94700	115100	89000	78000	73000	61800
TG3	–	179900	109400	–	121100	109300
TG4	–	326400	271700	–	199900	201700

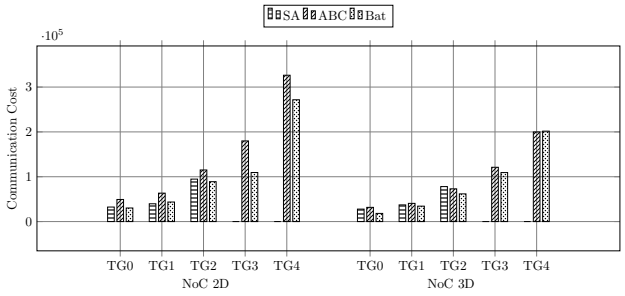


Figure 5. Result obtained in the synthetic benchmarks

The results obtained demonstrate that the Bat algorithm achieves better performance than simulated annealing (SA) across the three benchmarks considered, except for TG1 where SA delivers a slightly better outcome. However, when compared with the ABC algorithm, the five results in 2D are generally superior for ABC. In contrast, within the 3D architecture, the Bat approach outperforms both algorithms in most cases, again with the exception of TG1 where SA shows a slight advantage. These findings emphasize the competitiveness of the Bat algorithm in both 2D and 3D NoC environments.

We further extended the analysis by comparing the optimal results obtained with those reported in the literature from other state-of-the-art approaches. This two-stage evaluation not only demonstrates the efficiency and robustness of the proposed approach but also situates its performance within the current scientific landscape, thereby confirming its contribution to advancing efficient mapping strategies for 3D NoCs. The visual representations in Figure 4 enable a comparative analysis of the mapping outcomes achieved by SA in both 2D and 3D NoCs. The results summarized in Table 4 present the performance of the Bat mapping approach applied to the two topologies 2D and 3D NoC based on the synthetic benchmarks. Moreover, the graphical illustrations in Figures 6, 7, and 8 provide a clear comparison of the mapping outcomes obtained for both 2D and 3D NoCs with the VOPD, MPEG4, and MWD benchmarks, whereas for the PIP benchmark the results remain identical across all methods.

Table 4. Comparison results with other approaches in the 2D and 3D NoC

2D	VOPD	MWD	PIP	MPEG4	3D	VOPD	MWD	PIP	MPEG4
Bat	4345	1312	640	3735	Bat	4070	1180	640	3717
SA	4119	1184	640	3834	SA	4103	1138	640	3567
PSO	4141	1216	640	3757	NMAP 3D	4119	1184	640	3672
GBMAP	4217	–	–	3572	PSMAP 3D	4119	1120	640	3567
ONYX	4242	–	–	3612	DPSOMAP3D	4119	1120	640	3567
CGMAP	4300	–	–	3600	ILP 3D	4103	1120	640	3567
NMAP	4265	1344	640	3852	TSBM	4103	1120	640	3567
CHMAP	4167	1344	640	3852					
ILP	4119	1120	640	3567					
Castnet	4135	1280	–	3852					

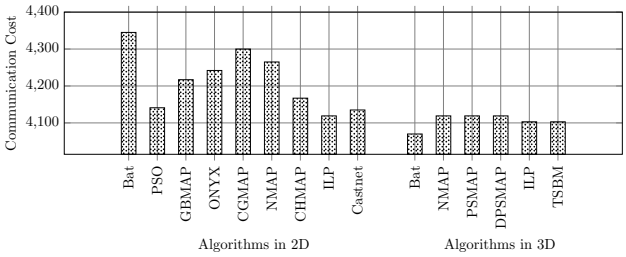


Figure 6. Result obtained in the VOPD benchmark

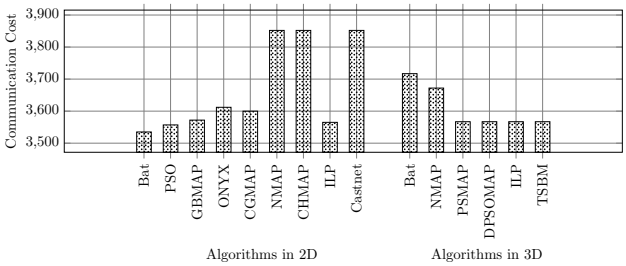


Figure 7. Result obtained in the MPEG4 benchmark

The comparison on real benchmarks shows that the Bat algorithm achieved better results for VOPD in 3D. However, for the other benchmarks, the outcomes reveal certain limitations. These limitations are mainly

due to the fact that a generalized configuration was applied uniformly across all benchmarks, without adapting the parameters to the specific requirements of each case. In particular, the number of iterations was fixed at 1500 for all experiments. This observation suggests that a tailored parameter tuning for each benchmark, especially in terms of iteration count and population size, could potentially lead to more significant performance improvements.

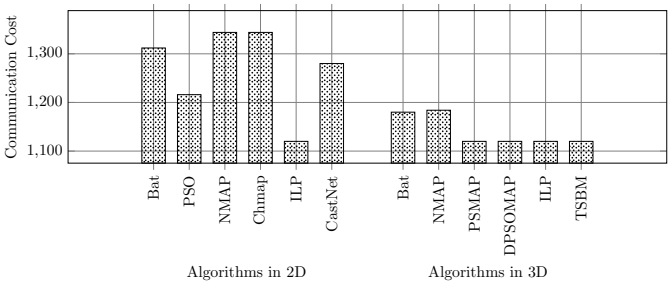


Figure 8. Result obtained in the MWD benchmark

7. CONCLUSION

This work presents an algorithm based on the Bat algorithm for application mapping in NoC architectures, targeting both 2D and 3D mesh topologies. The application mapping problem is known to be NP-hard and combinatorial in nature. To evaluate our approach, experiments were conducted using both real applications and synthetic benchmark core graphs, with a mono-objective optimization strategy where communication cost was the primary evaluation criterion. The experimental results demonstrate that the Bat algorithm achieves superior performance compared to other standard algorithms when appropriate parameter tuning is applied. However, for real benchmarks, only the VOPD case yielded the best performance, while the results on other benchmarks revealed some limitations. This is mainly due to the use of a generalized configuration across all benchmarks, whereas each benchmark requires its own specific configuration, particularly in terms of iteration count and parameter tuning. In future research, the proposed algorithm will be further extended to address a larger number of tasks and more complex optimization scenarios, thereby improving its scalability and robustness. Additionally, hybridization with other advanced metaheuristic techniques will be explored to enhance convergence accuracy and computational efficiency. Finally, the applicability of the algorithm will be investigated within wireless metwork-on-chip (WNoC) architectures, considering wireless-specific challenges such as latency, interference, and energy efficiency.

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
Maamar Bougherara	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Rafik Amara					✓					✓	✓			
Amina Guidoum					✓					✓	✓			

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

DATA AVAILABILITY

The authors confirm that the data supporting the findings of this study are available in [46].

REFERENCES

[1] J. Hu and R. Marculescu, "Energy-aware mapping for tile-based NoC architectures under performance constraints," in *Proceedings of the 2003 Asia and South Pacific Design Automation Conference*, 2003, pp. 233–239, doi: 10.1109/ASPDAC.2003.1195022.

[2] W. R. Davis *et al.*, "Demystifying 3D ICs: The pros and cons of going vertical," *IEEE Design & Test of Computers*, vol. 22, no. 6, pp. 498–510, 2005, doi: 10.1109/MDT.2005.136.

[3] M. Bougherara *et al.*, "IP assignment for efficient NoC-based system design using multi-objective particle swarm optimisation," *International Journal of Bio-Inspired Computation*, vol. 12, no. 4, pp. 203–213, 2018, doi: 10.1504/IJBIC.2018.096483.

[4] M. Bougherara *et al.*, "Core/task associations for efficient application implementation on network-on-chip," in *2018 International Conference on Computer and Applications (ICCA)*, 2018, pp. 18–22, doi: 10.1109/COMAPP.2018.8460231.

[5] M. Bougherara *et al.*, "Efficient application mapping onto three-dimensional network-on-chips using multi-objective particle swarm optimization," in *International Conference on Computational Science and Its Applications*, 2019, pp. 654–670, doi: 10.1007/978-3-030-24296-1_53.

[6] M. Bougherara *et al.*, "Routing in 3D NoCs using genetic algorithm and particle swarm optimization," in *International Conference on Computational Science and Its Applications*, 2023, pp. 601–613, doi: 10.1007/978-3-031-37105-9_40.

[7] L. Silva Junior *et al.*, "Efficient routing in network-on-chip for 3D topologies," *International Journal of Electronics*, vol. 102, no. 10, pp. 1695–1712, 2015, doi: 10.1080/00207217.2014.989545.

[8] M. Bougherara and R. Amara, "Routing using genetic algorithm in network on chips with 3D mesh topology," in *2023 International Conference on Computer and Applications (ICCA)*, 2023, pp. 1–5, doi: 10.1109/ICCA59364.2023.10401456.

[9] Q. Chen *et al.*, "An IP core mapping algorithm based on neural networks," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 29, no. 1, pp. 189–202, 2020, doi: 10.1109/TVLSI.2020.3033658.

[10] M. Bougherara *et al.*, "Application mapping onto 3D NoCs using differential evolution," in *Computational Science and Its Applications – ICCSA 2020*, 2020, pp. 89–102, doi: 10.1007/978-3-030-58808-3_8.

[11] P. K. Sahu and S. Chattopadhyay, "A survey on application mapping strategies for network-on-chip design," *Journal of Systems Architecture*, vol. 59, no. 1, pp. 60–76, 2013, doi: 10.1016/j.sysarc.2012.10.004.

[12] Y. Nalci *et al.*, "ILP formulation and heuristic method for energy-aware application mapping on 3D-NoCs," *The Journal of Supercomputing*, vol. 77, no. 3, pp. 2667–2680, 2021, doi: 10.1007/s11227-020-03365-0.

[13] S. Murali and V. De Micheli, "Bandwidth constrained mapping of cores onto NoC architectures," in *Design, Automation and Test in Europe Conference and Exhibition*, 2004, pp. 896–901, doi: 10.1109/DATE.2004.1269002.

[14] W. Shen *et al.*, "A new binomial mapping and optimization algorithm for reduced-complexity mesh-based on-chip network," in *Networks-on-Chip, NOCS 2007*, 2007, pp. 317–322, doi: 10.1109/NOCS.2007.5.

[15] S. Tosun *et al.*, "Application mapping algorithms for mesh-based network-on-chip architectures," *The Journal of Supercomputing*, vol. 71, no. 3, pp. 995–1017, 2015, doi: 10.1007/s11227-014-1348-x.

[16] C.-H. Cheng and W.-M. Chen, "Application mapping onto mesh-based network-on-chip using constructive heuristic algorithms," *The Journal of Supercomputing*, pp. 1–14, 2016, doi: 10.1007/s11227-016-1746-3.

[17] M. Janidarmian *et al.*, "Onyx: A new heuristic bandwidth-constrained mapping of cores onto tile based network on chip," *IEICE Electronics Express*, vol. 6, no. 1, pp. 1–7, 2009, doi: 10.1587/elex.6.1.

[18] A. Mehran *et al.*, "Spiral: A heuristic mapping algorithm for network on chip," *IEICE Electronics Express*, vol. 4, no. 15, pp. 478–484, 2007, doi: 10.1587/elex.4.478.

[19] M. Z. Dageleh and M. A. J. Jamali, "V-castnet3d: A novel clustering-based mapping in 3-D network on chip," *Nano Communication Networks*, vol. 18, pp. 51–61, 2018, doi: 10.1016/j.nancom.2017.11.002.

[20] F. Moein-darbari *et al.*, "CGMAP: A new approach to network-on-chip mapping problem," *IEICE Electronics Express*, vol. 6, no. 1, pp. 27–34, 2009, doi: 10.1587/elex.6.27.

[21] M. Tavanpour *et al.*, "GBMAP: An evolutionary approach to mapping cores onto a mesh-based NoC architecture," *Journal of Communication and Computer*, vol. 7, no. 3, pp. 1–7, 2010, doi: 10.17265/1548-7709/2010.07.003.

[22] P. K. Sahu *et al.*, "Application mapping onto mesh based network on chip using discrete particle swarm optimization," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 2, pp. 300–312, 2014, doi: 10.1109/TVLSI.2013.2240708.

[23] M. Bougherara *et al.*, "Application mapping onto network on chip using particul swarm optimisation with genetic algorithm GAPSO," in *2022 International Conference on Computer and Applications (ICCA)*, 2022, doi: 10.1109/ICCA56443.2022.10039542.

[24] M. Bougherara and R. Amara, "Application mapping onto network on chip using genetic algorithm and ant

colony optimisation," in *2023 International Conference on Computer and Applications (ICCA)*, 2023, pp. 1–6, doi: 10.1109/ICCA59364.2023.10401486.
[25] M. Bougherara and D. Messaoudi, "Genetic artificial bee colony for mapping onto network on chip GABC," in *Congress on Intelligent Systems*, 2022, doi: 10.1007/978-981-19-9225-4_11.
[26] M. Bougherara and Y. Imene, "Application mapping onto network on chip using cat swarm optimization," in *International Symposium on Intelligent Informatics*, 2022, pp. 441–453, doi: 10.1007/978-981-19-8094-7_34.
[27] M. Bougherara *et al.*, "DEMAP: Differential evolution mapping for network on chip optimization," *International Journal of Robotics and Automation (IJRA)*, 2023, doi: 10.11591/ijra.v12i4.pp394-404.
[28] C. Huang and D. Zhang, "Low-power mapping algorithm for three-dimensional network-on-chip based on diversity-controlled quantum-behaved particle swarm optimization," *Journal of Algorithms & Computational Technology*, vol. 10, no. 3, pp. 176–186, 2016, doi: 10.1177/1748301816649070.
[29] P. K. Hamedani *et al.*, "Exploration of temperature constraints for thermal aware mapping of 3D networks on chip," in *2012 20th Euromicro International Conference on Parallel, Distributed and Network-based Processing*, 2012, pp. 499–506, doi: 10.1109/PDP.2012.68.
[30] W. Jiawen *et al.*, "Energy-efficient mapping for 3D NoC using logistic function based adaptive genetic algorithms," *Chinese Journal of Electronics*, 2014, doi: 10.23919/CJE.2014.10851854.
[31] Y. Cui *et al.*, "Thermal and power aware task mapping on 3D network on chip," in *International Symposium on Integrated Circuits*, 2014, doi: 10.1016/j.compeleceng.2015.12.001.
[32] P. Kullu *et al.*, "An artificial bee colony based mapping method for three dimensional network-on-chip," in *2023 International Conference on Emerging Trends in Networks and Computer Communications (ETNCC)*, 2023, pp. 1–6, doi: 10.1109/ETNCC59188.2023.10284966.
[33] F. Asadzadeh *et al.*, "Thermal-aware application mapping using genetic and fuzzy logic techniques for minimizing temperature in three-dimensional network-on-chip," *The Journal of Supercomputing*, vol. 80, no. 8, pp. 11214–11240, 2024, doi: 10.1007/s11227-023-05869-x.
[34] M. Bougherara *et al.*, "Using simulated annealing for application mapping onto network on chip," in *2024 International Conference on Computer and Applications (ICCA)*, 2024, pp. 1–6, doi: 10.1109/ICCA62237.2024.10927822.
[35] B. N. K. Reddy and A. S. Kumar, "Evaluating the effectiveness of bat optimization in an adaptive and energy-efficient network-on-chip routing framework," *Journal of Parallel and Distributed Computing*, vol. 188, p. 104853, 2024, doi: 10.1016/j.jpdc.2024.104853.
[36] J. Li *et al.*, "Bat algorithm based low power mapping methods for 3D network-on-chips," in *National Conference of Theoretical Computer Science*, 2017, pp. 277–295, doi: 10.1007/978-981-10-6893-5_21.
[37] A. Alagarsamy and L. Gopalakrishnan, "MBA: A new cluster based bandwidth and power aware mapping for 2D NoC," in *2018 International Conference on Circuits and Systems in Digital Enterprise Technology (ICCSDET)*, 2018, pp. 1–4, doi: 10.1109/ICCSDET.2018.8821150.
[38] B. N. K. Reddy *et al.*, "Performance constrained multi-application network on chip core mapping," *International Journal of Speech Technology*, vol. 22, no. 4, pp. 927–936, 2019, doi: 10.1007/s10772-019-09636-3.
[39] X.-S. Yang, "A new metaheuristic bat-inspired algorithm," in *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*, 2010, pp. 65–74, doi: 10.1007/978-3-642-12538-6_6.
[40] X.-S. Yang and X. He, "Bat algorithm: Literature review and applications," *International Journal of Bio-Inspired Computation*, vol. 5, no. 3, pp. 141–149, 2013, doi: 10.1504/IJBIC.2013.055093.
[41] T.-K. Dao and T.-T. Nguyen, "A review of the bat algorithm and its varieties for industrial applications," *Journal of Intelligent Manufacturing*, vol. 36, no. 8, pp. 5327–5349, 2024, doi: 10.1007/s10845-024-02506-z.
[42] X.-S. Yang and A. Slowik, "Bat algorithm," in *Swarm Intelligence Algorithms*, 2020, pp. 43–53, doi: 10.1201/9780429422614-4.
[43] Z. Cui *et al.*, "Bat algorithm with principal component analysis," *International Journal of Machine Learning and Cybernetics*, vol. 10, no. 3, pp. 603–622, 2019, doi: 10.1007/s13042-018-0888-4.
[44] R. Y. M. Nakamura *et al.*, "Binary bat algorithm for feature selection," in *Swarm Intelligence and Bio-Inspired Computation*, 2013, pp. 225–237, doi: 10.1016/B978-0-12-405163-8.00009-0.
[45] R. P. Dick *et al.*, "TGFF: Task graphs for free," in *Proceedings of the Sixth International Workshop on Hardware/Software Codesign (CODES/CASHE'98)*, 1998, pp. 97–101, doi: 10.1109/HSC.1998.666245.
[46] M. Bougherara, "TGFF benchemarks NoC 2D 3D," doi: 10.13140/RG.2.2.20887.07842.

BIOGRAPHIES OF AUTHORS

Maamar Bougherara is an associate professor at the École Normale Supérieure of Kouba, Algeria. He holds a Ph.D. in computer science from the LIM Laboratory at Bouira University, Algeria. His current research focuses on networks-on-chip (NoC) and optimization algorithms. He also holds both an M.Sc. and an engineering degree in computer science from the University of Blida, Algeria. He can be contacted at bougherara.maamar@gmail.com.

Rafik Amara is an assistant professor at the Computer Science Department, Ecole Normale Supérieure de Kouba, Algiers. He obtained a computer engineering degree in 2001 from the University of Science and Technology (USTHB) in Algiers. After professional experience in the air navigation sector, he continued his studies to obtain in 2008 a master's degree in computer science, specializing in image processing and GIS. He is currently a doctoral student in the Image Processing and Radiation Laboratory (LTIR) at USTHB and more precisely in the GIS team. He works especially on linear networks such as air route networks and air trafc simulation and optimization. He can be contacted at rafik.amara@g.ens-kouba.dz.

Amina Guidoum an associate professor at the Higher School kouba, Algiers, Algeria, she holds master of science in computer science from the University of Sidi Bel Abbès; Doctor of Philosophy (Ph.D.) in network, architecture and multimedia from the University of Sidi Bel Abbès. Her research interests include images processing, multimedia systems, machine learning, and emerging technologies in multimedia. She can be contacted at guidoum_amina@hotmail.fr or amina.guidoum@g.ens-kouba.dz.